APLICACIÓN MÓVIL ANDROID PARA LA INTEGRACIÓN DE UNIVERSITARIOS CON DISCAPACIDAD AUDITIVA

Trabajo de Fin de Grado

Diego Domínguez Álvarez

Grado en Ingeniería Telemática
Universidad Carlos III

Tutor: Antonio de la Oliva Delgado
Director de proyecto: Enrique Mendoza Robaina

Septiembre 2015

Resumen

Este proyecto trata del diseño e implementación de un sistema de aplicaciones Android más un servidor, el cual permita ayudar a resolver el problema de la atención dividida, que presentan los estudiantes con discapacidad auditiva en las clases de universidad. Este problema afecta a una parte importante de los estudiantes discapacitados que llegan a la universidad, e intenta proponer una alternativa a la actual costosa solución, que son los intérpretes de la lengua de signos.

El principal objetivo de este proyecto es que el estudiante pueda prestar atención a la vez al discurso del profesor, a la pizarra y a tomar apuntes. Para ello se desarrollarán dos aplicaciones Android y un servidor. Una aplicación para el profesor que sea capaz de grabar tanto el vídeo de la clase como la voz del profesor. Otra aplicación para los alumnos que permita seguir la clase, leer la transcripción del profesor, hacer preguntas y otros muchos extras más.

El presente documento tiene como finalidad la presentación del proceso completo llevado a cabo para el análisis, diseño e implementación del proyecto. Además se tratarán las principales tecnologías utilizadas, como son la plataforma Android y diversas tecnologías de servidor. También se explicarán las metodologías de trabajo seguidas, junto a una planificación y presupuesto del proyecto. Por último, se incluyen una serie de pruebas de funcionamiento y las conclusiones obtenidas al realizar el proyecto.

Abstract

This project is about the design and implementation of a system composed by two Android applications and a server, that will be capable of helping to solve the problem of divided attention, which many deaf students face when they are in class at the university. This problem affects a considerable portion of university students that are disabled, and tries to propose an alternative solution to sign language interpreters, which is very expensive.

The main objective of this project is that the student can pay attention at the same time to the speech of the teacher, the blackboard and take notes. In order to achieve this, we will develop two Android applications and a server. An application for the teacher that will be capable of recording the video of the class as well as the voice of the professor. An application for the students that will allow to follow the class, read the transcript of the teacher, ask questions and many more extras.

This whole document has the aim of describing the process taken to analyze, design and implement this project. Also there will be explained the main technologies used in this project, as it is the Android platform and diverse server technologies. In addition, the working methodology followed in this project will be explained, together with a planning and a budget list of the project. Finally, there will be included some real case testing and the conclusions obtained from the completion of this project.

Tabla de contenidos

Lista de figuras

Lista de tablas

Chapter 1

Introduction

In this chapter we will present a general overview of the project, the motivation, its main goals, the socio-economic environment, regulatory framework and finally, we will present the organization of the whole document.

1.1 General overview

This project growths from an internship that the author of this memory, Diego Dominguez, had from November 2014 to May 2015 with Telefónica and the collaboration of Fundación Adecco, under the program Talentum Startups. Here he is proposed to apply a technological solution to the problem of divided attention that deaf students face during their university classes, under the supervision of Enrique Mendoza.

The main idea of the solution is that the student can attend at the same time to the speech of the teacher, the blackboard and take notes. In order to achieve that it will be developed a system called Breaking Sound Barriers, from now on BSB, composed by two Android applications and a server.

There will be developed an application for the professor, called BSB Server, that will be in charge of recording the video of the class as well as transcribing the audio of the professor. An application for all the students of the class, called BSB App, that will allow to follow the class, read the transcription of the professor, make questions and a lot more extras. Finally, a server devoted to the client-server communication between both apps and the data stored.

1.2 Motivation

The main motivation of this project is to provide a technological solution, which will solve the problem of divided attention that affects to deaf students during classes at the university.

This problem affects with a reasonable impact to many students, that are not capable of following voice indications from the professor at the same time they write notes from the blackboard or projectors. This happen because they usually have to pay attention to a third element, their sign language interpreter that normally goes with them to class or for the most advanced ones, they pay attention to the lips, which needs a visual of the professor face a hundred percent of the time, which is certainly not possible for a professor during a normal class.

Another motivation that arrives with this project is the reduction of cost. Universities spend most of their disable people budget in hiring language sign interpreters. One of this interpreters not only has to be there with the student during the classes, which is a tough exercise to be transcribing for the length of one to two hours of class non-stop, what it makes more difficult is to understand the topic of the speech given that we are in an university class discussing from engineering to law topics. The language of signs is a pretty simple language that mainly understands actions and verbs without tense, and many words are agreed between the interpreter and the student. Words like impedance or financial derivative are not common words in this language, and are difficult to explain, which leads to extra work for the interpreter as well as the student.

One last thing that pushes this project is the aim of integrating deaf people. It is always difficult for a person that has to be in class separate from the rest of mates with an interpreter. This project wants to create a tool that everyone in class will want to use for a better following of the class with no distinction of what are your physical disabilities.

1.3 Goals

Taking into account the motivations that we have just stated, we can say that our **main goal** is to create a system of applications and a server that together can solve the problem of divided attention that deaf students have when they are in class at the university. Together with this main goal, go these **primary objectives**:

- Create an app for the professor that can record the voice and the video of the class during the streaming, as well as an easy management of the different courses that the professor may have.
- Another app for the students that will receive the streaming, and be capable of taking notes and asking questions to the teacher at the same time.
- Develop a server that will manage the connections between the professor and the students, control the access to the system and manage the data stored in the database.

Apart from the main goals we can state some **secondary goals** that will make our project richer and more outstanding:

- The solution must use the most recent open source technologies available in the market, that will guarantee an easy development as well as a low cost avoiding licensing.
- We want to be the tool as integrative as possible, meaning that as well as helping deaf students, other students will want to use it.
- The apps must be straightforward to control and to set up, especially for the professor. We do not want them to loose time during classes.
- We want that the student app becomes a study tool, giving them the option to take notes, use a repository, a forum and more extras.
- The project has to have a simple implementation and deployment, assuring that universities can adapt this solution inn a proper way.

1.4 Socio-economic environment

This project is focused on helping people with hearing disabilities. It wants to become a tool for the society, we do not want only to help deaf people to follow their classes at the university, we want them to be a part of this society. In order to achieve it, this project has been developed in a way it is as inclusive as possible so that any disabled person can feel they are treated the same as any other people at class.

The project is also centered on reducing the cost that nowadays universities expend on helping disabled students. They do not usually have a big budget for this purposes,

and the most part of it goes, in this case, to hire sign language interpreters. For the better understanding it has been made a budget of the cost of this project (see Chapter 4), here we will explain the cost savings that this solution brings.

1.5 Regulatory framework

As this is a software project we rely on licensing to use others source code. On the application side, all the source code and libraries used in relation to Android are under the Apache Software License, Version 2.0 [1]. For the server side, we choose to use Play Framework, also under Apache 2.0, and for its database MySQL from Oracle, under version 2 of the GNU General Public License. Like other free software licenses, these licenses allow us, users of this software, to use the software for any purpose, to distribute it, to modify it, and to distribute modified versions of the software, under the terms of the license, without concern for royalties. Applying to both sides, it is guaranteed the protection of all personal data provided when using this application, and in compliance with the provisions of Law of Data Protection. There are no other technical or legal regulations that restrict or apply to the problem.

1.6 Document organization

This section is devoted to explain briefly the organization of the different chapter in this memory:

- **Chapter 1:** Introduction. In this chapter we will present a general overview of the project, the motivation, its main goals, the socio-economic environment, regulatory framework and finally, we will present the organization of the whole document.
- **Chapter 2:** State of the art. In this chapter we will see some technologies that are related to the project, in order to explain everything that is necessary for the understanding of the project. First, we will see what in the Android platform; a brief introduction, its history, architecture, main APIs (Application Programming Interfaces) and the development environment. Then we will see the technologies used in the server, such us the framework used, its database and the characteristics of the client-server connections.

- **Chapter 3:** Analysis, design and implementation of the system. In this chapter we will perform an analysis of the requirements of the project to the latter design the solution and finally explain how it it implemented. We will also make some test to measure the performance of the system.
- **Chapter 4:** Budget and planning. In this chapter we will show the estimate cost of developing and implementing this project together with the organization that has been taken to develop this project. We will also cover the techniques used for collaborative work during this project.
- **Chapter 5:** Conclusions and future work. In this chapter we will make some conclusion about the work done in this project and propose some future work that can be applied to improve it.

At the end of the memory there are three appendixes with extra information about the project; an extended English summary, a user manual and a deployment manual.

Capítulo 2

Estado del Arte

En este capítulo veremos algunas tecnologías que se utilizan durante el desarrollo de la memoria, para un mejor entendimiento del proyecto. Primero veremos la plataforma Android; una breve introducción, su historia, arquitectura, las principales APIs y su entorno de desarrollo. Tras esto, veremos las tecnologías utilizadas en el servidor, como el framework utilizado, la base de datos y las características de las comunicaciones cliente-servidor.

2.1 Android

Para la realización del presente proyecto se ha utilizado la plataforma Android, por lo que serán explicadas sus principales características al lector durante la siguiente sección. En la siguiente figura podemos ver el logo de Android, Figura 2.1.

Fig. 2.1 Logo de Android [2]

2.1.1 Introducción a Android

Android [3] es el sistema operativo de Google. Está basado en el **Kernel de Linux**, que utiliza para controlar servicios del núcleo del sistema como la gestión de procesos, memoria o seguridad. Se ha desarrollado principalmente para su uso en pantallas táctiles, como pueden ser smartphones y tablets. Actualmente su uso también se ha expandido a televisiones (Android TV), coches (Android Auto) y relojes (Android Wear). Inicialmente se lanzó en Octubre de 2008 sin ninguna versión en clave, no sería hasta la tercera versión con Android 1.5 "Cupcake" cuando recibiría su primer nombre.

Una de las principales características de este sistema operativo es que cuenta con una gran comunidad de desarrolladores, que aportan a su crecimiento. Esto es debido a que Android se distribuye como una plataforma de código abierto con el nombre de AOSP (Android Open Source Project), bajo la licencia Apache 2.0, lo que significa que cualquier persona puede modificarlo o distribuirlo [1]. Aun así, la mayoría de dispositivos Android vienen con una mezcla de **software libre y propietario** licenciado por Google. La fundación en 2008 de la OHA (Open Handset Alliance), un consorcio de empresas de hardware, software y telecomunicaciones, hizo que el desarrollo de Android no sólo formase parte de Google, sino que permitió incluir software propietario a otras compañías. Podemos observar en la Figura 2.2 algunas de sus compañías.

Fig. 2.2 Compañías en la OHA [4]

Otro motivo por el cual los desarrolladores eligen Android es porque la propia Google ofrece gratuitamente su entorno de desarrollo, SDK (Software Development Kit), antiguamente como plugin al entorno de desarrollo Eclipse y actualmente como una suite completa

en Android Studio, basando la programación en el lenguaje **Java**. Conjuntamente se ofrece también un emulador del sistema operativo, funcional en todos los sistemas operativos, que ofrece una alternativa si no se dispone de un dispositivo real Android. En este emulador se permite hacer pruebas de la interfaz gráfica a diferentes tamaños de pantalla y resolución, además de debug en tiempo real.

Android cuenta actualmente con la mayor tienda de aplicaciones [5], el Google Play Store con cerca de 1.3 millones de aplicaciones y más de 50 billones de descargas. Esto también es debido a que cuenta con un numero mayor de desarrolladores que otras plataformas, unos 400 mil. Además, el Google I/0 de 2015 reveló que el numero de dispositivos Android activos superó el billón [6], lo que se traduce en **8 de cada 10 dispositivos móviles en el mundo son Android**. A continuación, en la Figura 2.3 podemos observar el número de aplicaciones en los diferentes mercados de aplicaciones.

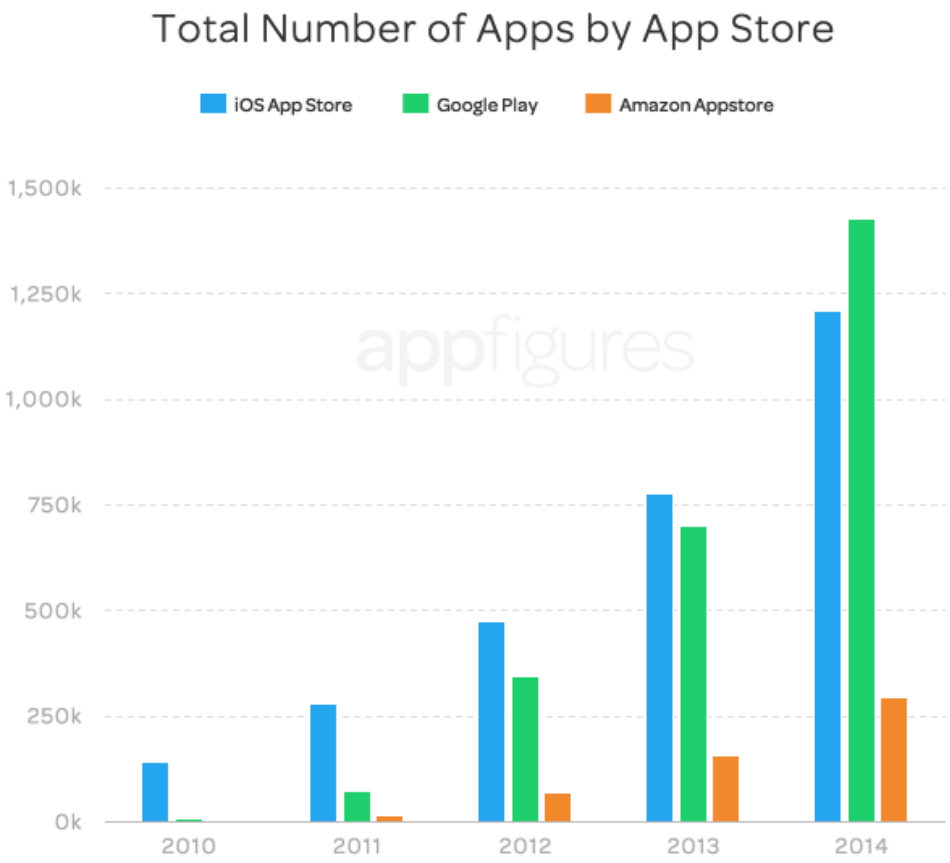


Fig. 2.3 Número total de aplicaciones por tienda [5]

2.1.2 Historia de Android

La historia de Android comienza en 2003 con sus primeros desarrollos por la empresa Android Inc., la cual fue comprada por Google en 2005 [7]. No es hasta 2007 cuando se revela la primera beta de Android, y tras esta la primera versión comercial, Android 1.0 gracias al desarrollo de Google y la OHA. Actualmente la versión que mas usuarios tiene es la 4.4 "KitKat", siendo la versión 5.1 la mas reciente "Lollipop". En la siguiente figura, podemos observar la distribución de las versiones de Android a lo largo de los años, Figura 2.4.

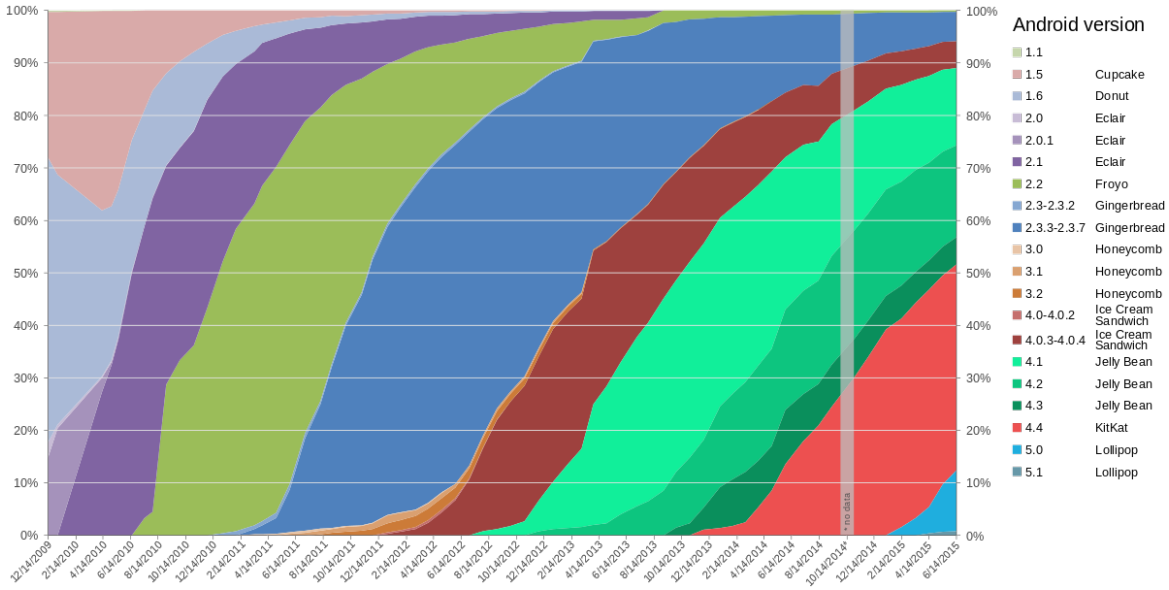


Fig. 2.4 Distribución histórica de las versiones de Android [7]

A continuación veremos los principales avances que han ido incorporando cada versión de Android:

- **Cupcake (1.5):** Lanzado el 30 de abril de 2009. Es la primera versión que recibe un nombre. El primer dispositivo Android, el HTC Dream, incorporó las siguientes características principales:
  - Soporte para Widgets.
  - Grabación y reproducción en formatos MPEG-4 y 3GP.
  - Agregada la opción de auto-rotación.
  - Habilidad de subir vídeos a YouTube.

- **Donut (1.6):** Lanzado el 15 de septiembre de 2009, incorporó las siguientes características principales:
  - La entrada de texto y voz es mejorada para incluir historial de favoritos, contactos y la web.
  - Soporte de la tecnología CDMA/EVDO, 802.1x, VPNs y un motor text-to-speech.
  - Soporte para resoluciones de pantalla WVGA.
  - Se incluye la herramienta de desarrollo GestureBuilder, un framework para gestos ampliado.
- **Eclair (2.0–2.1):** Lanzado el 26 de octubre de 2009, incorporó las siguientes características principales:
  - Soporte de sincronizaron con múltiples cuentas de correo.
  - Soporte Bluetooth 2.1.
  - Mejoras en la cámara incluyendo soporte de flash, zoom digital, modo escena, balance de blancos, efecto de colores y enfoque macro.
  - Nuevo navegador con soporte para HTML5, nueva interfaz con imágenes en miniatura de marcador y zoom de toque-doble.
- **Froyo (2.2–2.2.3):** Lanzado el 20 de mayo de 2010, incorporó las siguientes características principales:
  - Funcionalidad de anclaje de red por USB y Wi-Fi hotspot.
  - Actualización de la aplicación Market (actual Play Store) que incluye actualizaciones automáticas.
  - Soporte para Adobe Flash.
  - Soporte para pantallas de alto número de PPI (Pixels Per Inch) (320 ppi), como 4" 720p.
- **Gingerbread (2.3–2.3.7):** Lanzado el 6 de diciembre de 2010, incorporó las siguientes características principales:
  - Se actualiza la interfaz de usuario con mejoras de velocidad y simpleza.
  - Soporte para NFC (Near Field Communication).
  - Mejor regulación del uso de batería en aplicaciones de administración.

  - Soporte nativo para sensores de giroscopio y barómetro.

- **Honeycomb (3.0–3.2.6):** Lanzado el 22 de febrero de 2011, fue la primera versión dedicada solo para tablets y no apta para teléfonos Android. Incorporó las siguientes características principales:

  - Soporte optimizado para tablets aportando una nueva interfaz de usuario holográfica.
  - Inclusión de la barra de sistema, proporcionando acceso rápido a notificaciones, estados y botones de navegación, disponible en la parte inferior de la pantalla.
  - Añadida la barra de acción (Action Bar), agregando acceso a opciones contextuales, navegación o widgets en la parte superior de la pantalla.
  - Simplificación de la multitarea, permitiendo saltar de una aplicación a otra rápidamente a través de una interfaz con instantáneas de las aplicaciones en curso.

- **Ice Cream Sandwich (4.0–4.0.4):** Lanzado el 19 de octubre de 2011, incorporó las siguientes características principales:

  - La Action Bar y la barra de sistema ya disponibles también para teléfonos Android.
  - Posibilidad de cerrar aplicaciones en segundo plano.
  - Captura de pantalla integrada.
  - Google Chrome se estable como nuevo navegador con pestañas.

- **Jelly Bean (4.1–4.3.1):** Lanzado el 27 de junio de 2012, fue el primer gran cambio de diseño en la interfaz gráfica de Android. Incorporó las siguientes características principales:

  - La pantalla de bloqueo se actualiza, incluyendo soporte para widgets y añadiendo acceso directo a la cámara.
  - Se homogeneiza el diseño de la interfaz para todos los dispositivos, con botones centrados en la pantalla, la barra de sistema en la parte superior de la pantalla, y una pantalla de inicio con un dock y el menú de aplicaciones centrado.
  - Inclusión de la tecnología OpenGL ES 3.0.
  - Optimización del sistema operativo para funcionar en dispositivos con bajas especificaciones, tan solo 512MB de RAM.

- **KitKat (4.4–4.4.4, 4.4W–4.4W.2):** Lanzado el 31 de octubre de 2013. Se caracteriza por ser la versión más estable de Android y que más usuarios utilizan. Incorporó las siguientes características principales:
  - Transparencias en la barra de estado y barra de navegación.
  - WebViews basadas en el motor de Chromium.
  - Nuevo marco de transiciones y efectos visuales.
  - Introducción del modo inmersivo, el cual oculta la barra de estado y de navegación para visualizar las aplicaciones en pantalla completa.
- **Lollipop (5.0–5.1.1):** Lanzado el 3 de noviembre de 2014, destacó principalmente por el gran cambio de diseño, bautizado Material Design. Incorporó las siguientes características principales:
  - Accesos directo a aplicaciones y notificaciones desde la pantalla de bloqueo.
  - Cambio de Android Runtime de Dalvik a ART, pasando de una compilación JIT (Just In Time) a AOT (Ahead OF Time).
  - La pantalla de actividades pasa de mostrar aplicaciones a tareas.
  - Creación del Project Volta, para las mejoras de la vida de la batería.

También es interesante ver como se distribuye actualmente los tamaños de los dispositivos, en la Figura 2.5, con una tendencia a dispositivos en trono a las 4 pulgadas y con una densidad alta de píxeles.

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	4.1%						4.1%
Normal		7.6%	0.1%	39.9%	19.8%	15.9%	83.3%
Large	0.4%	4.8%	2.2%	0.6%	0.6%		8.6%
Xlarge		3.1%		0.3%	0.6%		4.0%
Total	4.5%	15.5%	2.3%	40.8%	21.0%	15.9%	

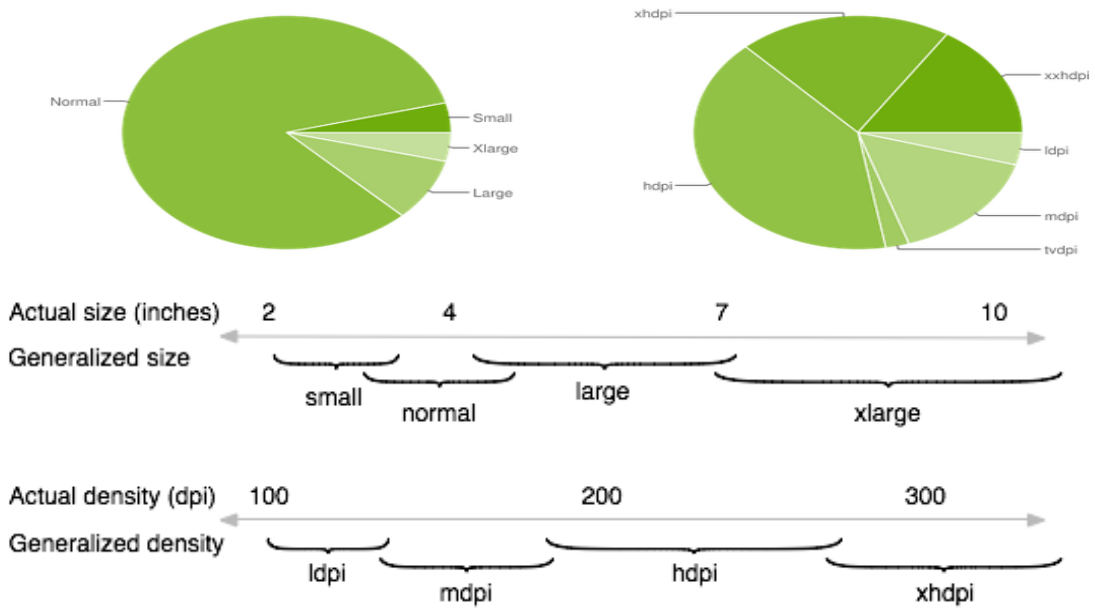


Fig. 2.5 Distribución actual de los tamaños de los dispositivos [8, 9]

2.1.3 Arquitectura

La arquitectura básica de Android es muy parecida a la de **Unix**, donde las aplicaciones interactúan con el hardware del dispositivo usando el Kernel, y en este caso a través de las librerías y el run time. Como podemos observar en la figura 2.6, la arquitectura básica está compuesta por 5 bloques, **aplicaciones, framework de aplicaciones, librerías, Android Run Time y Kernel de Linux** . En la Figura 2.6 se muestra un esquema de la arquitectura de software de Android. A continuación se explicaran cada uno de estos bloques.

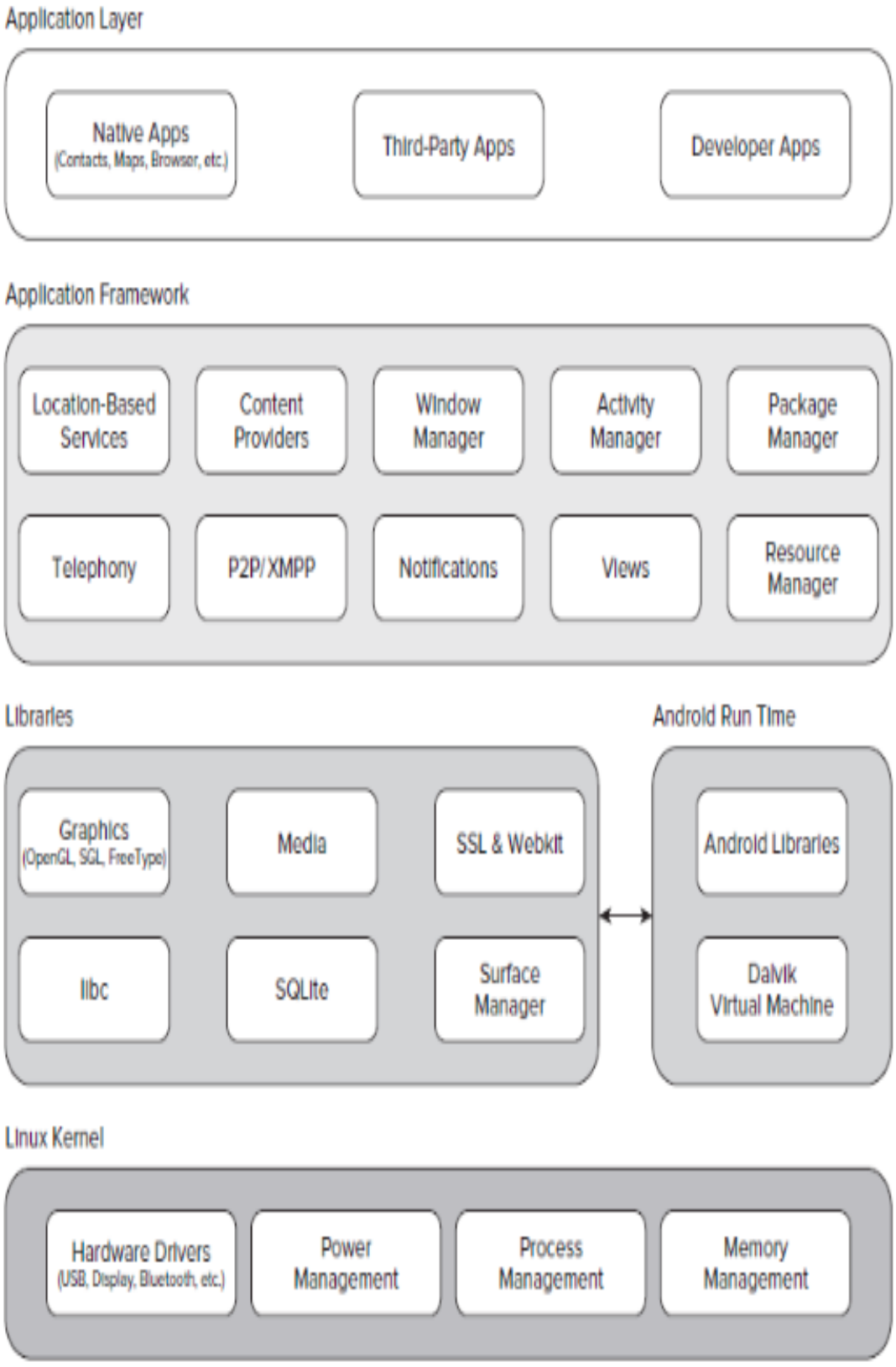


Fig. 2.6 Pila de software de Android [10]

Aplicaciones

El bloque de aplicaciones es la capa más alta de la arquitectura de Android, con la que el usuario puede directamente interactuar. Estas consisten en todas las operaciones básicas relacionadas con la telefonía, como son los servicios de mensajería, telefonía, contactos o navegación web, además de todas las que el usuario quiera instalar, como son juegos o apps. Esta capa interactúa con el resto de bloques para conseguir dar el servicio que pretende proporcionar la aplicación al usuario.

Las aplicaciones Android están escritas en el lenguaje de programación **Java** [11], y su código es compilado gracias al SDK en un APK (Android Package), un archivo que permite la instalación de la aplicación en un dispositivo. Una vez instalada la aplicación, esta contiene una serie de características dentro del sistema Android:

- Cada aplicación vive en su propio entorno limitado de seguridad, ya que cada aplicación es un usuario diferente de Linux en el sistema operativo Android.
- Por defecto cada aplicación corre en su propio proceso de Linux, teniendo cada una su propio ID de usuario de Linux. Lo que permite al sistema dar permisos de acceso a ficheros a la aplicación.
- Cada proceso tiene su propia maquina virtual, por lo que el código está aislado de otras apps.

Framework de aplicaciones

Este bloque se encarga de acercar a los programadores todos los servicios que ofrece el sistema operativo Android, tanto hardware como software, a través de unas APIs. Además, Android está desarrollado de tal forma que incluye una serie de mecanismos que permiten a los programadores acceder a funcionalidades de otras aplicaciones del sistema, es decir, la arquitectura de las aplicaciones están compuestas por **componentes** que pueden compartir entre si, siempre y cuando se cumplan unas normas de seguridad y así lo haya permitido el desarrollador. Los componentes que componen una aplicación Android son:

- **Actividades:** Una actividad representa una pantalla con una interfaz gráfica. Normalmente, una actividad refleja una determinada actividad llevada a cabo por una

aplicación. Una aplicación suele estar compuesta por más de una pantalla e incluirá más de una actividad. Por ejemplo, una aplicación de email podrá contener dos actividades diferentes, una para mostrar correos y otra para componerlos. Cualquier otra aplicación del sistema podrá lanzar estas actividades, si está permitido. Por ejemplo, la aplicación de la cámara podrá lanzar la actividad de componer un email para componer un nuevo email con una foto adjunta.

- **Servicios:** Un servicio es un componente que se ejecuta en segundo plano y no ofrece interfaz gráfica. Está pensado para realizar operaciones que tengan una gran carga y duren cierto tiempo o que necesiten ejecutar trabajo en un proceso remoto. Por ejemplo, un servicio puede destinarse a reproducir música en segundo plano mientras el usuario está en otra aplicación, o también puede estar realizando una comprobación a través de Internet sin interferir al usuario mientras interactúa con otra actividad.

- **Broadcast receivers:** Un broadcast receiver es un componente que responde a anuncios difundidos por el sistema, que pueden ser tanto basados en cambios de estado de software como de hardware. Una aplicación puede emitir sus propios broadcast específicos, y aunque no van acompañados de una interfaz gráfica, pueden aparecer como notificaciones para alertar del evento ocurrido. Broadcast receivers son cambios en el nivel de batería, la pantalla esta encendida o un archivo acaba de ser descargado para que otra actividad o servicio sea avisada de que esta disponible para su uso.

- **Content provider:** Un content provider gestiona un conjunto compartido de datos de las aplicaciones. Gracias a ellos puedes almacenar datos en el sistema, en una base de datos SQlite, en la web o cualquier otro tipo de almacenamiento persistente que tu aplicación pueda acceder. Mediante el content provider cualquier aplicación, si se tiene permisos, puede hacer consultas a esos datos, editarlos o generar nuevos. Por ejemplo, leer el contenido de una nota de la aplicación de notas.

Junto con estos componentes, Android también cuenta con dos gestores de recursos para el mejor funcionamiento del sistema:

- **Gestor de notificaciones:** Lleva a cabo la recopilación de las notificaciones del sistema y las muestra de una forma visual en la barra de estado del teléfono.

- **Gestor del ciclo de vida:** Es el componente encargado de manejar el ciclo de vida de las actividades y el paso de unas a otras entre aplicaciones del sistema. Android maneja las actividades como una pila, siendo siempre la nueva actividad que se lance la situada

en la cima de la pila, quedando la anterior detrás en la pila. Para la administración del ciclo de vida existen una serie de métodos relacionados a eventos, que se definen en una actividad. Por ejemplo, al evento de finalizar una aplicación sería conveniente en el método onDestroy() cerrar cualquier conexión a Internet que esté establecida o guardar los cambios. La Figura 2.7 muestra la relación y transición entre estados de una actividad.

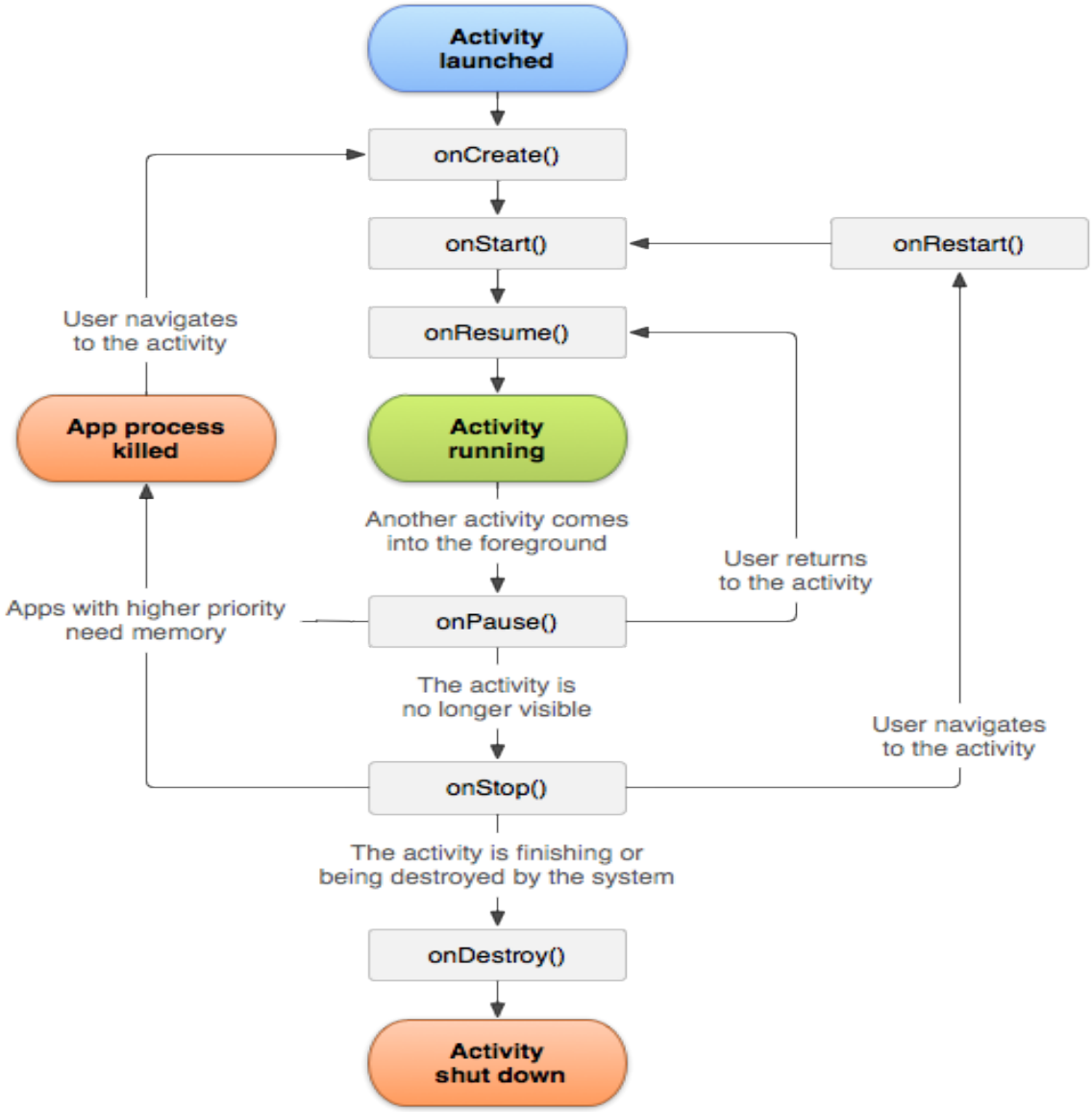
Librerías

En Android están incluidas una serie de librerías implementadas en los lenguajes C y C++, que son usadas para utilizar los componentes del sistema operativo. Estas librerías son accesibles a los programadores gracias al framework de aplicaciones antes mencionado gracias a las APIs. El hecho de que se usen estos lenguajes es debido a que ofrecen un mayor rendimiento que Java, aún así los propios desarrolladores pueden incluir sus propias librerías. Algunas de las librerías más importantes son: librerías del sistema C, librerías multimedia, el gestor de superficie, WebKit, OpenGL y SQLite.

Android Run Time

Como se mencionó antes, Android se ejecuta sobre el Kernel de Linux. Las aplicaciones de Android están escritas en Java, por lo que se ejecutan sobre una **máquina virtual**. Cada aplicación es un proceso y se ejecuta en su propia instancia de la máquina virtual Dalvik, una tecnología de código abierto. Esta máquina virtual ejecuta ficheros de tipo .dex, los cuales han sido optimizados para hacer el menor uso posible de recursos de memoria y CPU del sistema. Un compilador de Java transforma las clases en este tipo de ficheros. Con la última versión del sistema, Lollipop, el Run Time del sistema cambia de Dalvik a ART [13] incluyendo las siguientes mejoras:

- Compilación AOT, que mejora el rendimiento de las aplicaciones, reduciendo el tiempo de ejecución, pero aumentando el tamaño de los archivos.
- Mejorado el GC (Garbage Collection) para una mejor respuesta de las aplicaciones durante su ejecución.
- Mejoras para desarrolladores y debugging.



Method	Description	Killable?	Next
`onCreate()`	Called when the activity is first created. This is where you should do all of your normal static set up: create views, bind data to lists, etc. This method also provides you with a Bundle containing the activity's previously frozen state, if there was one. Always followed by `onStart()` .	No	`onStart()`
`onRestart()`	Called after your activity has been stopped, prior to it being started again. Always followed by `onStart()`	No	`onStart()`
`onStart()`	Called when the activity is becoming visible to the user. Followed by `onResume()` if the activity comes to the foreground, or `onStop()` if it becomes hidden.	No	`onResume()` or `onStop()`
`onResume()`	Called when the activity will start interacting with the user. At this point your activity is at the top of the activity stack, with user input going to it. Always followed by `onPause()` .	No	`onPause()`
`onPause()`	Called when the system is about to start resuming a previous activity. This is typically used to commit unsaved changes to persistent data, stop animations and other things that may be consuming CPU, etc. Implementations of this method must be very quick because the next activity will not be resumed until this method returns. Followed by either `onResume()` if the activity returns back to the front, or `onStop()` if it becomes invisible to the user.	**Pre-** `HONEYCOMB`	`onResume()` or `onStop()`
`onStop()`	Called when the activity is no longer visible to the user, because another activity has been resumed and is covering this one. This may happen either because a new activity is being started, an existing one is being brought in front of this one, or this one is being destroyed. Followed by either `onRestart()` if this activity is coming back to interact with the user, or `onDestroy()` if this activity is going away.	**Yes**	`onRestart()` or `onDestroy()`
`onDestroy()`	The final call you receive before your activity is destroyed. This can happen either because the activity is finishing (someone called `finish()` on it, or because the system is temporarily destroying this instance of the activity to save space. You can distinguish between these two scenarios with the `isFinishing()` method.	**Yes**	*nothing*

Fig. 2.7 Ciclo de vida de las actividades de Android [12]

Kernel de Linux

Android está basado en la versión del **Kernel de Linux 2.6**, para dar los servicios base del sistema como son el manejo de memoria a bajo nivel, manejo de procesos, pila de red, seguridad y gestión de drivers. Además, el Kernel de Linux actúa como una capa de abstracción entre el hardware y el resto del software del sistema operativo.

2.1.4 Application Programming Interfaces (APIs)

Como se ha comentado anteriormente, las APIs son el conjunto de clases, paquetes, atributos en XML y librerías que dan acceso a los desarrolladores a todas las partes del sistema operativo Android. A continuación veremos las APIs más importantes y utilizadas para el desarrollo de las aplicaciones Android de este proyecto y su funcionamiento:

- **Android.Content:** Contiene las clases para el acceso y la publicación de datos en el dispositivo, dando acceso a los content providers. Existen tres categorías principales: Content Sharing, Package Management y Resource Management. La clase principal utilizada en este proyecto es *SharedPreferences*, que da acceso al almacenamiento persistente que posee cada app, normalmente asociado a guardar las configuraciones del usuario.
- **Android.hardware.Camera:** La clase *Camera* se utiliza para configurar los ajustes de captura de imagen, iniciar o detener la vista previa, tomar fotos, y recuperar los marcos para la codificación de vídeo. Esta clase es un cliente para el servicio de la cámara, que gestiona el hardware real de la cámara. Las clases principales usadas en este proyecto son *Parameters*, para la obtención de la configuración de la cámara y *PreviewCallback*, para recuperar copias de los fotogramas de la vista preliminar de la cámara a la vez que se están mostrando.
- **Android.Bluetooth:** Proporciona clases que administran la funcionalidad Bluetooth, tales como la exploración de dispositivos, la conexión con los dispositivos, y la gestión de la transferencia de datos entre dispositivos. La API de Bluetooth es compatible con "Bluetooth Classic" y "Bluetooth Low Energy". Este proyecto utiliza principalmente los Broadcast Receivers que manda esta API para detectar la conexión de dispositivos Bluetooth, y una vez reconocidos, redirigir el micrófono y audio del dispositivo Android al dispositivo Bluetooth.

- **Android.Speech:** Esta API es usada para dar servicio a las motores de lenguaje de Text-to-Speech y Speech-to-Text. El motor Text-to-Speech sintetiza el discurso del texto para su reproducción inmediata o para crear un archivo de sonido, utilizada en este proyecto para mandar mensajes de sonido predeterminados. Para el Speech-to-Text se utiliza el motor SpeechRecognizer, utilizado para recibir notificaciones cuando un evento relacionado ocurre, es decir, se detecta el comienzo o fin del discurso, si el motor de reconocimiento está listo, se ha procesado un resultado correctamente o erróneamente.
- **Android.Gms:** Google Mobile Services es la API de Google que permite mejorar la experiencia de usuario mediante Google Play Services. A través de ella se pueden acceder a los servicios de mapas, localización, fitness y social que ofrece Google. Es importante destacar que Google Mobile Services es diferente de Android y no comparten el mismo tipo de licencias. Esta API no se utilizará durante el desarrollo del proyecto, pero es importante destacarla, ya que da acceso a todos los servicios de Google, presentes en la gran mayoría de dispositivos Android.

2.1.5 Entorno de desarrollo

A la hora de desarrollar una aplicación en Android existen diversas alternativas a la hora de elegir el entorno de desarrollo. Hasta Noviembre de 2014 la opción recomendada por Google era utilizar una serie de plugins para el entorno de desarrollo Eclipse. Actualmente, con el lanzamiento de **Android Studio**, este se convierte en la herramienta de apoyo que mejor integra y pone a disposición del programador los servicios necesarios para el desarrollo de aplicaciones Android. Entre ellos destacan asistentes de desarrollo de interfaces gráficas, asistente de modelado de imágenes a diferentes PPI, integración con sistemas de control de versión, debuggin mediante la herramienta LogCat, comunicación con el dispositivo mediante ADB, monitorización de procesos y recursos del dispositivo, grabación de la pantalla y muchos otros servicios que ayudan al desarrollo de las aplicaciones.

La organización de un proyecto de Android se muestra en la siguiente Figura 2.8:

- **/manifests/AndroidManifest.xml:** Fichero manifiesto (XML) del módulo (app móvil).
- **/java/<paquete>/*.java:** Ficheros fuente de código java (MainActivity.java).
- **/res/:** Recursos utilizados por la aplicación (imágenes, de texto, etc.).

- **/res/drawable:** Imágenes y otros recursos multimedia.
- **/res/layout:** Definición de layouts en XML de la aplicación.
- **/res/menu:** Definición de los menús en XML de la aplicación.
- **/res/values:** Varios ficheros XML para definir strings, colores, estilos, etc.

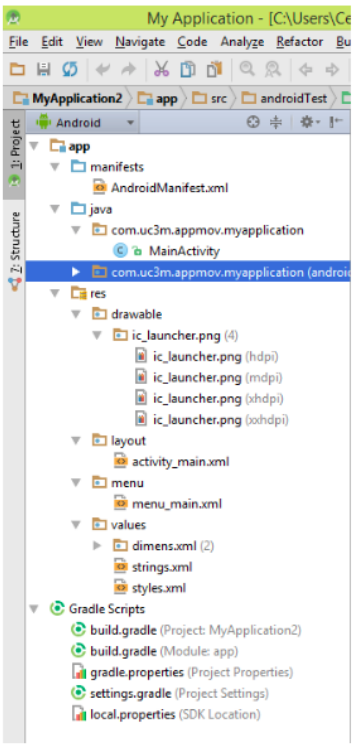


Fig. 2.8 Estructura de un proyecto de Android [14]

2.2 Tecnologías del servidor

Para el desarrollo del proyecto se ha necesitado la implantación de un servidor para manejar las comunicaciones entre los clientes de las aplicaciones Android y la gestión de sus recursos. A continuación se explicarán diversas tecnologías que se han aplicado en la implementación del servidor.

2.2.1 Servidor

Un servidor [15] es cualquier proceso informatizado que comparte diferentes recursos hacia uno o varios procesos de un cliente. Los servidores siguen la arquitectura de cliente-servidor, esto es, la tarea del servidor es ser capaz de atender a una petición de un cliente y devolverle una respuesta a corde. La comunicación entre ellos se suele realizar a través de la red en Internet, como podemos observar en la Figura 2.9. Aunque también es posible desplegarse en local, en el propio ordenador personal. Por motivos de seguridad, es recomendable el uso de maquinas dedicadas "servidores" para el despliegue de estos programas, ya que pueden ser accedidos desde cualquier parte de la red.

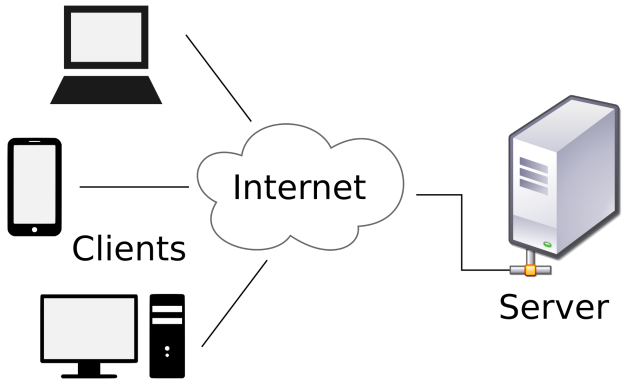


Fig. 2.9 Arquitectura cliente-servidor [16]

Tipos

Existen varios tipos de servidores. A continuación se indican algunos de los más comunes:

- **Servidor de archivos:** Se encarga de almacenar diferentes tipos de archivos para distribuirlos entre otros clientes de la red.
- **Servidor de correo:** Gestiona operaciones de correo electrónico, como almacenar, enviar, enrutar los mensajes de los clientes de la red.
- **Servidor de acceso remoto:** Ofrece el acceso a distancia de dispositivos localizados en una red remota.
- **Servidor web:** Ofrece diferentes archivos Web como documentos HTML, imágenes y otros tipos de archivos a petición de los clientes de la red.

- **Servidor de base de datos:** Provee de acceso a las operaciones con bases de datos.

Según su función dentro de la red se pueden clasificar en:

- **Servidor dedicado:** Es aquel que se dedica únicamente a atender las solicitudes de los clientes en la red.
- **Servidor no dedicado:** Es aquel que dedica su actividad parte hacia a los clientes y parte a sus operaciones locales.

Web Application Frameworks

Ahora que conocemos lo que es un servidor, en el desarrollo de este proyecto se ha elegido utilizar un servidor web dedicado, y así atender las peticiones de los clientes Android. Para que esta tarea sea más sencilla, se ha decido utilizar un **marco de desarrollo o framework**.

Un WAF [17] (Web Application Framework) es un framework de software diseñado para ayudar al desarrollo de aplicaciones web, páginas web, servicios web y recursos web. Un framework pretende aligerar la sobrecarga asociada a las tareas comunes de un desarrollo web, como pueden ser la gestión de sesiones, el acceso a base de datos y encaminamiento de rutas, para lo que se proporcionan bibliotecas y se promueve la reutilización de código ya implementado para realizar tales tareas. Esto permite al desarrollador no tener que reimplementar las mismas funciones para cada aplicación web que desarrolle.

Aunque cada framework es diferente, las características principales de un WAF son:

- **Estructuras de datos persistentes:** La principal característica de una aplicación web es construir páginas web en base a su información almacenada, que es generada de forma dinámica a partir de datos persistentes.
- **Manejo de sesiones y autenticación de usuarios:** Las aplicaciones web normalmente necesitan diferenciar usuarios anónimos de los usuarios registrados en el sistema. Para ello se crean sesiones y mecanismos de autenticación, que permiten la persistencia de los datos a través de las páginas, acorde con el usuario.
- **Seguridad:** Ciertas partes del servicio web pueden estar restringidas a cierto tipo de usuarios. Es por etso, que se ofrecen servicios de login y de permisos por usuario.

- **Caching:** Para mejorar el rendimiento, los desarrolladores web suelen almacenar en caché información que no tiene que ser regenerada en cada petición.

- **Interfaz de administrador:** Es común que en web dinámicas exista un perfil de administrador que permita configurar el servidor y sus datos.

La mayoría de frameworks de aplicaciones están basados en el **patrón de diseño MVC** [18] (Modelo–Vista–Controlador). Este es un patrón de diseño de arquitectura de software. Se basa en separar a partes distintas la lógica de negocio de la interfaz de usuario, ayudado por un módulo que gestiona los eventos y comunicaciones entre ellos. El patrón se basa en las ideas de reutilización de código y separación de conceptos, muy ligados a los conceptos de framework, ayudando al mejor desarrollo de aplicaciones y su mantenimiento.

MVC se construye a base de tres componentes: **el modelo**, representa la información del sistema y gestiona su acceso y modificación, **el controlador**, es el encargado de responder a los eventos del usuario y mandar peticiones al modelo y **la vista**, representa el modelo y su información al usuario mediante una interfaz de usuario que pueda interactuar.

El flujo de control que sigue el patrón suele seguir el esquema de la Figura 2.10:

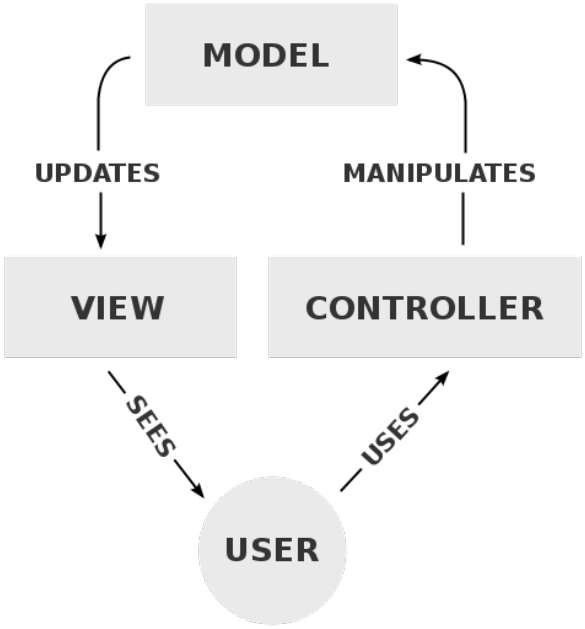


Fig. 2.10 Arquitectura MVC [18]

1. El usuario interactúa con la interfaz gráfica.
2. El controlador recibe la acción descrita por el usuario a través de la interfaz gráfica, generalmente usando un handler o callback.
3. El controlador accede al modelo, atendiendo a la petición del usuario y hace las modificaciones oportunas en él.
4. La vista obtiene los datos cambiados en el modelo y procede a generar la nueva interfaz gráfica al usuario, acorde al nuevo modelo modificado.
5. La interfaz de usuario vuelve a estar lista para interactuar con el usuario.

Play Framework

De entre los muchos WAF que existen se decide utilizar Play framework para desarrollar el servidor de este proyecto, concretamente la versión 2.3.

Play [19] es un WAF de código libre basado en Java y Scala, que mantiene una arquitectura MVC. Está diseñado para ayudar a **optimizar la productividad** del desarrollador mediante el uso de convención sobre configuración, recarga de código en tiempo real y visualización de errores en el navegador. Además incluye una gran extensión de librerías, plantillas y comunidad de desarrolladores, que hacen que los proyectos tengan un despliegue funcional y robusto en poco tiempo. Las principales diferencias de Play con otros frameworks son:

- Play es completamente RESTful, no mantiene estados por conexión.
- Incluye un completo test de unidad integrado.
- La arquitectura es modular.
- Play es capaz de dar servicio a largas peticiones de una manera asíncrona.
- Es completamente operable con las APIs de Java y Scala.
- Analizador sintáctico de JSON y XML.
- Soporte para los IDE Eclipse y IntelliJ IDEA.
- Compatibilidad nativa con WebSockets.

2.2.2 Bases de datos

Una base de datos [20] es el **conjunto de datos almacenados**, entre los que se encuentran esquemas, tablas, vistas, consultas y otros objetos, que siguen una estructura dentro de un mismo contexto. Esta estructura intenta modelar algún aspecto de la realidad de tal manera que ofrece su procesamiento, como puede ser encontrar los teléfonos de una persona dentro de una agenda de contactos.

Los SGBD (Sistemas Gestores de Bases de Datos) son aplicaciones de software encargados de manejar la base de datos, dando una interfaz de usuario con la que interactuar, herramientas con las que analizar sus datos y su administración. Un SGDB destaca por su capacidad de abstracción de la información, la facilidad de manejo en las transacciones y la consistencia de los datos. Ejemplos de SGDB son MySQL, SQL Server o Oracle Database.

Existen varios tipos de bases de datos atendiendo a su modelo de administraron de datos, siendo el más popular el modelo relacional, representado por el lenguaje SQL (Structured Query Language):

- **Jerárquicas:** Se organizan a modo de árbol invertido, donde hay un nodo padre o raíz de información del que cuelgan nodos hijo. Muy eficientes para aplicaciones con gran cantidad de datos, pero incapaces de representar la redundancia de información.
- **De red:** Es un modelo parecido al jerárquico, en el que un nodo hijo puede tener varios padres, lo cual soluciona la redundancia de datos.
- **Transaccionales:** Se caracterizan por la velocidad de gestión de datos en envío y recepción, aunque son poco comunes.
- **Relacionales:** Es el modelo más usado para gestionar datos dinámicamente. La forma de almacenamiento de datos es en tablas y se accede a su información mediante consultas en el lenguaje SQL.
- **Multidimensionales:** Es un modelo similar al relacional, diferente a modo de concepto. Se caracteriza por tener una única tabla o columna por dimensión.
- **Orientadas a objetos:** Se caracteriza por estar basado en los lenguajes de programación orientados a objetos, con propiedades como el polimorfismo, la herencia o la encapsulación. En este modelo se guardan todos los elementos de un objeto.

- **Documentales:** Permiten indexar y guardar la información de forma completa como un documento. Por ejemplo XML, JSON, Microsoft Office Document o PDF.
- **Deductivas:** Son bases de datos que permiten hacer deducciones mediante inferencias. Su almacenamiento esta basado en hecho o reglas. También son llamadas bases de datos lógicas debido a que su gestión sigue la lógica matemática.

Para el desarrollo de este proyecto se decide utilizar una base de datos relacional MySQL por las características del proyecto y las opciones que facilita este modelo. Como podemos observa en la Figura 2.11:

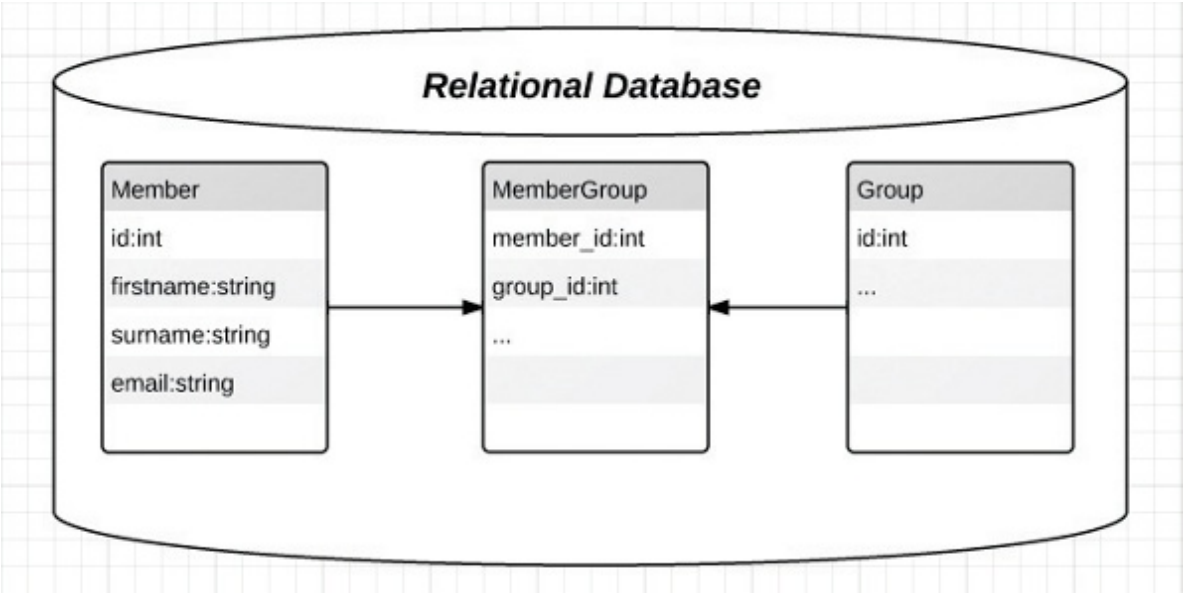


Fig. 2.11 Diagrama de tablas de una base de datos relacional [21]

Bases de datos relacionales

Las bases de datos relacionales [22] siguen el modelo relacional, el más usado para implementar bases de datos planificadas. Este tipo de bases de datos se caracterizan porque las **tablas estén relacionadas entre sí** en algunos de sus atributos. Las tablas que componen la base de datos no pueden estar repetidas, es decir, no hay dos con un mismo nombre. Estas están compuestas por un registro de filas y columnas. El almacenamiento de datos puede estar restringido por estos registros, por ejemplo, un campo numérico de tamaño máximo de 9 dígitos solo podrá almacenar números de tal longitud y no cadenas de texto. Para la relación entre campos de diferentes tablas se utilizan dos tipos de claves:

- **Clave primaria:** La clave primaria especifica un campo o combinación de ellos de forma única a cada fila de una tabla. No puede haber dos filas en una tabla con la misma clave primaria en ese campo. Se utilizan para relacionar con las claves foráneas de otras tablas.

- **Clave foránea:** La clave foránea es la referencia a la clave primaria en otra tabla, determinando la relación entre las dos tablas. Las claves foráneas no tienen por que ser únicas, pero solo referenciadas a una sola clave primaria. Por ejemplo, una contacto puede tener varios teléfonos personales, pero un teléfono personal no puede corresponder a varios contactos.

Existe un tercer tipo de clave, la clave índice, que son creados a partir de cualquier combinación de campos de una tabla. Esta sirve para facilitar un acceso más rápido a los datos mediante consultas, ya que se filtran los registros por esos campos.

La estructura de las bases de datos relacionales se compone de los siguientes elementos:

- Nombre de las tablas.
- Nombre de cada campo (columna).
- Tipo de datos de cada campo.
- Tabla a la que pertenece cada campo.

MySQL

MySQL [23] es un sistema de base de datos que utiliza el modelo relacional. Se caracteriza por ser un sistema muy popular que ofrece un acceso a base de datos multihilo y multiusuario. MySQL se distribuye mediante una licencia dual, una licencia GNU GPL de código libre, y mediante licencia específica para productos privativos.

Para acceder a una base de datos se pueden utilizar diversos lenguajes de programación como C, C++, Java, etc. Es común utilizar el estándar OBDC (Open DataBase Connectivity) para el acceso a la base de datos, en el caso de Java se utiliza **JDBC**. Este tipo de conexiones son de gran seguridad, ya que se requiere una identificación para cada conexión a la base de datos.

Algunas de las características por las que destaca MySQL son:

- Amplia utilización del lenguaje SQL.
- Accesible desde diversas plataformas y sistemas operativos.
- Ofrece diferentes mecanismos de almacenamiento.
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto.

2.2.3 Conexiones Cliente-Servidor

Nuestros clientes Android realizarán peticiones al servidor, y este le responderá con una serie de datos para que pueda continuar con la ejecución. Para esta conexión cliente-servidor una serie de datos serán enviados a través de la red. Para ello se modelan dos tipos de comunicaciones, una API tipo REST basada en HTTP y una comunicación por WebSockets. La información transmitida se encapsula en el formato JSON. A continuación se describen los mecanismos utilizados:

REST

REST (REpresentational State Transfer) [24] es un modelo de abstracción para crear APIs de aplicaciones de una manera estandarizada. Típicamente en el desarrollo de aplicaciones web, como el servidor de este proyecto, la arquitectura REST tiene un fin en el protocolo HTTP. Una de las características claves de REST, es la de ofrecer un **servicio escalable** que proporcione el mínimo encabezamiento en la comunicación final con el cliente. Una ejemplo de este tipo de comunicaciones se puede observar en la Figura 2.12. REST se apoya en una serie de fundamentos clave:

- Un protocolo **sin estado**, cada petición HTTP es independiente y suficiente para comprender toda la información en la comunicación cliente-servidor.

- Los recursos de información se acceden mediante **operaciones bien definidas**, en el caso de HTTP son POST, GET, PUT y DELETE, requeridas para la persistencia de los datos.
- Una **dirección única** para cada recurso del sistema, en este caso una única URI.
- La transición entre estados de la aplicación y el acceso a recursos debe ser mediante hipermedios, es decir, la navegación entre diferentes recursos REST es posible sin necesidad de estructuras externas.

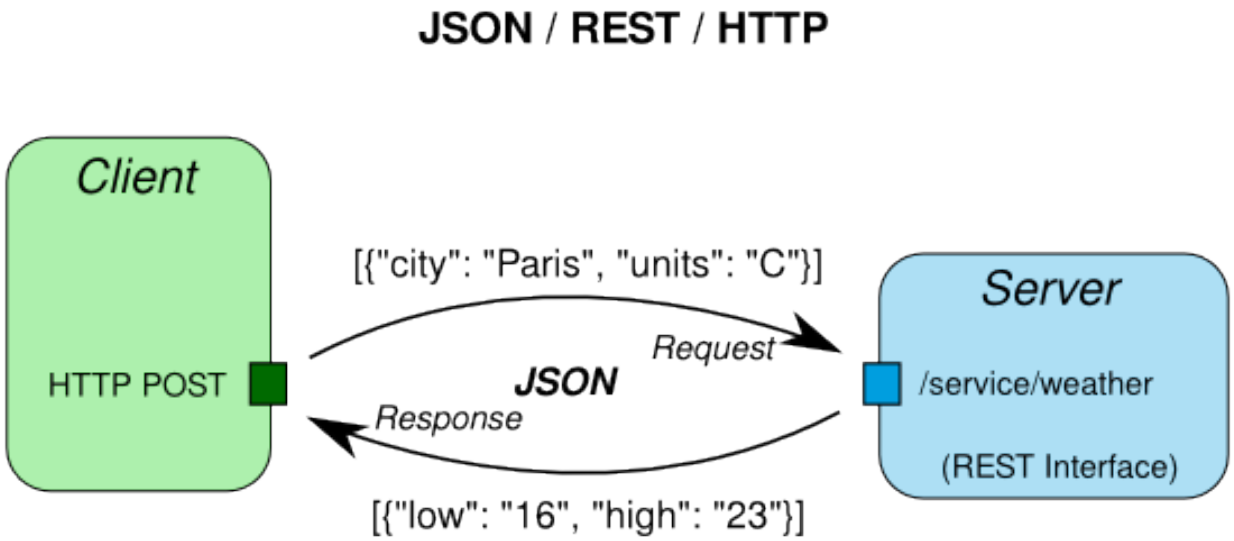


Fig. 2.12 Ejemplo de comunicación REST sobre HTTP con contenido JSON [25]

HTTP [26] es el protocolo utilizado bajo el modelo REST, siendo el encargado de mandar documentos a través de la red. HTTP es un protocolo sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores, lo que hace encajar perfectamente con la filosofía de REST.

En HTTP existen dos tipos de rol, el cliente y servidor. El cliente suele mandar peticiones y el servidor las responde. Estas respuestas vienen acompañadas de un código de respuesta y texto por norma general. Los códigos de respuesta más habituales son:

- **200 OK:** Indica que la petición fue correcta.
- **201 Created:** La respuesta indica que un recuso fue creado correctamente a partir de una petición PUT o POST.
- **400 Bad Request** La petición fue creada incorrectamente y no pude ser validada o no cumple el formato.
- **404 Not Found** Esta respuesta indica que el recurso solicitado no ha sido encontrado.
- **401 Unauthorized** Indica que es necesaria una previa autorización para acceder a ese recurso.
- **500 Internal Server Error** Esta respuesta indica que el proceso en servidor ha fallado y reporta un mensaje de error.

En HTTP los recursos están localizados mediante URLs. Cada URL identifica un tipo de recursos, como puede ser un documento HTML. Para el acceso a estos recursos se utilizan diferentes verbos o métodos HTTP:

- GET: Esta petición reclama una representación del recurso identificado por la URL.
- PUT: Se utiliza esta petición para crear o actualizar un recurso referenciado por una URL.
- DELETE: Al contrario que PUT, se usa cuando se quiere borrar un recurso.
- POST: Una petición POST sirve cuando quieres indicar que haya un proceso en el servidor de los datos enviados.

WebSockets

El protocolo WebSocket [27] proporciona una **comunicación bidireccional** adaptada al uso entre clientes y servidores en una aplicación web. El objetivo de este protocolo es crear una tecnología, que proporcione un mecanismo a las aplicaciones web que necesiten una comunicación bidireccional entre el cliente y servidor. El protocolo esta definido por el IETF en el RFC6455, mientras que la API de WebSockets esta siendo desarrollada por el W3C [28] [29]. El protocolo consiste en la comunicación por un canal TCP, en el que tras

una petición de negociación se intercambian tramas de mensajes. La Figura 2.13 muestra un ejemplo de como se establece una conexión mediante WebSockets.

El protocolo consiste de tres partes:

1. **Petición de negociación:** Una vez que las peticiones de negociación son enviadas y se acuerdan los puertos TCP por donde la comunicación se va a establecer se puede empezar a enviar tramas de mensajes.
2. **Transferencia de información:** Por el canal de transmisión establecido pueden mandarse mensajes encapsulados en tramas. Cada trama posee mensajes del mismo tipo de datos.
3. **Petición de cierre de canal:** Para terminar una comunicación por WebSockets se mandará una trama de cierre, que responderá la otra parte. Así se garantiza que antes de cerrar el canal todos los datos son transmitidos. Es seguro mandar simultáneamente las tramas de cierre.

Actualmente en el lado del cliente están implementados los WebSockets en Mozilla Firefox 11, Google Chrome 16, Internet Explorer 10 y Safari 6.

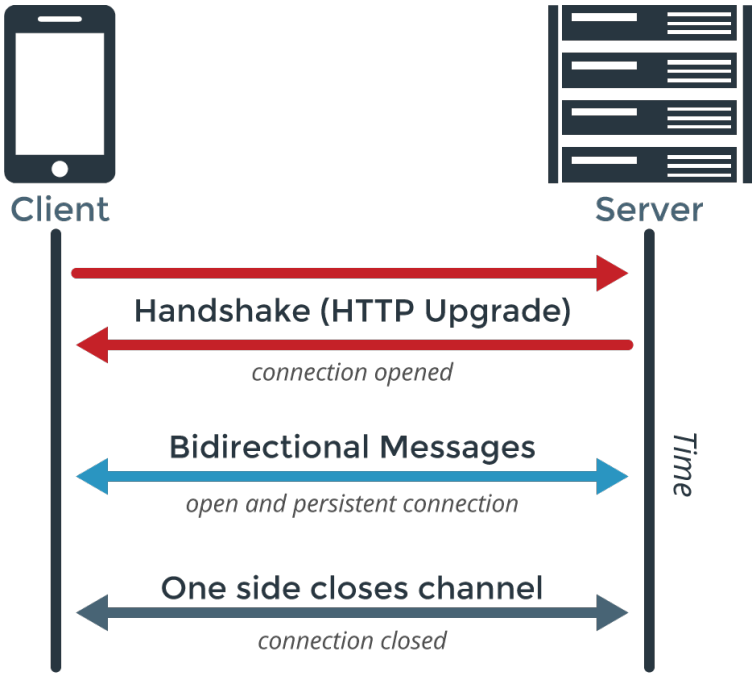


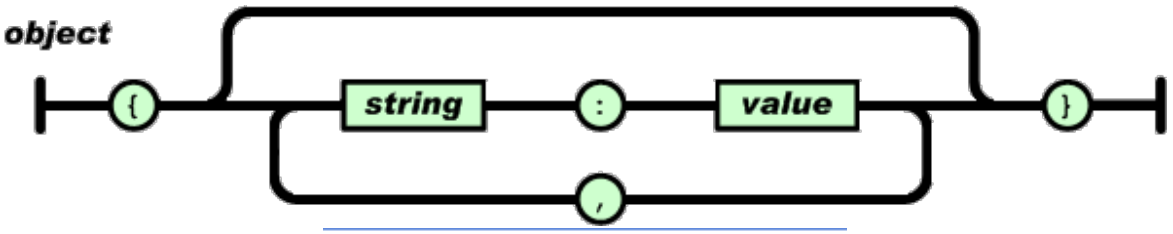
Fig. 2.13 Diagrama de una comunicación por WebSockets [30]

JSON

JSON [31] (JavaScript Object Notation) es un formato de **codificación muy ligero** para el intercambio de mensajes. JSON es una codificación sencilla, por lo que es más fácil de generar y analizar sintácticamente para las aplicaciones. Está basado en la programación en lenguaje JavaScript, aunque es independiente de cualquier lenguaje y puede usar usado en cualquiera de ellos. Esto hace JSON sea una manera ideal de intercambio de datos entre lenguajes. Un ejemplo de un objeto codificado en formato JSON se puede ver en la Figura 2.14

Los objetos JSON son construidos en base de dos estructuras:

1. Una colección de **parejas nombre-valor**, en otros lenguajes llamado objetos, struct, hash table, etc.
2. Una **lista ordenada de valores**, generalmente en otros lenguajes arrays, vectores, listas, etc.




```
[
    {
        "id": 1,
        "name": "Budweiser",
        "type": "Lager",
        "country": "USA"
    }, {
        "id": 2,
        "name": "Iron City",
        "type": "Pilsner",
        "country": "USA"
    }
]
```

Fig. 2.14 Objeto JSON [31]

Capítulo 3

Análisis, diseño e implementación del sistema

En este capítulo se hará un análisis de los requisitos del proyecto, se explicará el diseño del sistema y la implementación a nivel de código. Para cada apartado se explicarán tanto las dos aplicaciones Android como el servidor de forma separada. Finalmente se hará una exposición de las pruebas de funcionamiento del sistema en un entrono real.

3.1 Análisis

En esta sección se expondrán los diferentes requisitos del proyecto. Para definir estos requisitos se empleará el estándar IEEE 830 [32]. Con este estándar se definen unas tablas de fácil creación y comprensión que permiten definir los requisitos de un proyecto.

3.1.1 Requisitos de cliente

La descripción de los requisitos del cliente se hará mediante la representación de dos tipos de tablas diferentes, todas ellas siguiendo el estándar IEEE 830. Los requisitos de función indican las funciones necesarias para cumplir los propósitos requeridos para el sistema. Los requisitos de restricción indican las limitaciones del sistema. Con ellos se pretenderá cumplir los objetivos principales y secundarios del proyecto.

El formato que tendrán las tablas será el seguido por la Tabla 3.1.

Nombre			ID
Dependencias		Prioridad	
Descripcion			

Tabla 3.1 Tabla modelo requisitos

- **Nombre:** Nombre inequívoco del requisito a modo de resumen de este.
- **ID:** Identificador inequívoco del requisito. Estará formado por dos partes, el tipo de requisito y el código único de requisito.
- **Prioridad:** Representa la importancia o prioridad de implementación del requisito. Este campo tomará valores de 1 a 3, siendo 1 el máximo de importancia.
- **Dependencias:** Indica si este requisito depende de la implementación de otro requisito determinado.
- **Descripción:** Incluye una descripción breve de lo que incluirá el requisito.

Requisitos de función

A continuación se incluyen los requisitos de función:

1. Requisitos generales (RFG):

Vista de identificación de usuarios			**RFG-01**
Dependencias	-	Prioridad	1
Descripcion	Nada más entrar en la aplicación deberá de encontrarse un formulario donde introduciendo correo y contraseña el usuario pueda autenticarse.		

Tabla 3.2 RFG-01 Vista de indentificación de usuario

Vista principal			**RFG-02**
Dependencias	-	Prioridad	1
Descripcion	Tras autenticarse y entrar en el sistema, se debe mostrar una vista de inicio con consejos de cómo utilizar la aplicación y demás caracteristicas individuales de cada aplicación.		

Tabla 3.3 RFG02- Vista principal

Vista configuración perfil			**RFG-03**
Dependencias	-	Prioridad	1
Descripcion	Debe de existir una vista de perfil donde el usuario pueda modifcar sus datos personales, avatar, contraseña…		

Tabla 3.4 RFG-03 Vista configuración perfil

Vista foro			**RFG-04**
Dependencias	-	Prioridad	1
Descripcion	La vista de foro incluirá un foro de discusión para las asignaturas que el profesor o alumno hayan asistido para posteriror debate de las clases. Se permitirá a alomnos como a profesores crear hilos de discusión.		

Tabla 3.5 RFG-04 Vista foro

Vista streaming			**RFG-05**
Dependencias	-	Prioridad	1
Descripcion	La vista de streaming debe permitir la actividad principal de la aplicación, grabar el video y audio del profesor y transmitirlo a los alumnos.		

Tabla 3.6 RFG-05 Vista streaming

Salida del sistema			**RFG-06**
Dependencias	RFG-03	Prioridad	3
Descripcion	La aplicación debe permitir salir de la aplicación a un usuario para que pueda logearse otro.		

Tabla 3.7 RFG06- Salida del sistema

Borrado de datos			**RFG-07**
Dependencias	RFG-03	Prioridad	3
Descripcion	La aplicación debe permitir el borrado de historial y datos que almacene del usuario en el sistema, como apuntes, fotos y asignaturas recientes.		

Tabla 3.8 RFG-07 Borrado de datos

2. Requisitos de la aplicación del alumno (RFA):

Listado de actividad reciente			**RFA-01**
Dependencias	RFG-02	Prioridad	2
Descripcion	En la vista principal se debe acompañar la actividad reciente del alumno en las asignaturas de una forma resumida.		

Tabla 3.9 RFA-01 Listado de actividad reciente

Listado de asignaturas abiertas			**RFA-02**
Dependencias	RFG-05	Prioridad	1
Descripcion	Al entrar a la vista de streaming se deben de mostrar las sesiones abiertas de las asignaturas.		

Tabla 3.10 RFA-02 Listado de asignaturas abiertas

Búsqueda de sesiones			**RFA-03**
Dependencias	RFG-05	Prioridad	3
Descripcion	En la vista de streaming se debe poder hacer una búsqueda de las sesiones que hay abiertas por nombre del profesor o asignatura.		

Tabla 3.11 RFA-03 Búsqueda de sesiones

Visionado de streaming			**RFA-04**
Dependencias	RFG-05	Prioridad	1
Descripcion	En la vista de streaming, dentro de una sesión el alumno debe de disponer de un lugar para visionar el streaming de vídeo y los subtítulos.		

Tabla 3.12 RFA-04 Visionado de streaming

Tomar notas			**RFA-05**
Dependencias	RFG-05	Prioridad	1
Descripcion	En la vista de streaming, dentro de una sesión el alumno debe de disponer de un lugar para tomar apuntes mientras visiona el streaming.		

Tabla 3.13 RFA-05 Tomar notas

Preguntas al profesor			**RFA-06**
Dependencias	RFG-05,RFG-04	Prioridad	2
Descripcion	En la vista de streaming, dentro de una sesión el alumno debe de disponer de un lugar para poder enviar preguntas al profesor . Además, un sistema de votos para las preguntas ya formuladas y así decidir cual llegará al profesor y se mostrara como otro hilo del foro.		

Tabla 3.14 RFA-06 Preguntas al profesor

Capturas de pantalla			**RFA-07**
Dependencias	RFG-05	Prioridad	3
Descripcion	En la vista de streaming, dentro de una sesión el alumno podrá realizar capturas de pantalla con un acceso directo.		

Tabla 3.15 RFA-07 Capturas de pantalla

Vista repositorio			**RFA-08**
Dependencias	RFG-05	Prioridad	1
Descripcion	La vista de repositorio debe de incluir todos los apuntes tomados por el alumno, capturas de pantalla y discurso del profesor; propiamente ordenados por asignatura y sesión.		

Tabla 3.16 RFA-08 Vista repositorio

Configuración de subtítulos			**RFA-09**
Dependencias	RFG-05, RFG-03	Prioridad	3
Descripcion	En la vista de perfil debe de existir una opción para configurar la manera en que aparecen los subtíutlos.		

Tabla 3.17 RFA-09 Configuración de subtítulos

3. Requisitos de la aplicación del profesor (RFP):

Listado de asignaturas			**RFP-01**
Dependencias	RFG-01	Prioridad	1
Descripcion	En la vista pricipal se deben mostrar las asignaturas que el profesor en cuestión a configurado para impartir clase.		

Tabla 3.18 RFP-01 Listado de asignaturas

Configuración streaming			**RFP-02**
Dependencias	RFG-01, RFG-05	Prioridad	1
Descripcion	El profesor puede acceder a crear una nueva sesion de su asignatura desde la vista principal. De una manera sencilla podrá configurar la privacidad de la sesión. Una vez en la pantalla de streaming, deberá ser fácil configurar los dispositivos bluetooth y empezar a hacer streaming de la clase.		

Tabla 3.19 RFP-02 Configuración de streaming

Creación asignatura			**RFP-03**
Dependencias	RFG-01	Prioridad	1
Descripcion	El profesor puede acceder a crear una nueva asignatura desde la vista principal de una forma intuitiva, en la que puede configurar su nombre y foto de fondo.		

Tabla 3.20 RFP-03 Creación asignatura

Configuración asignatura			**RFP-04**
Dependencias	RFG-03	Prioridad	1
Descripcion	El profesor puede modificar las características de la asignatura una vez creada desde la vista de perfil.		

Tabla 3.21 RFP-04 Configuración asignatura

Grabado de audio			**RFP-05**
Dependencias	RFG-05	Prioridad	1
Descripcion	La aplicación debe de ser capaz desde la vista de streaming de grabar el audio que proviene del dispositivo bluetooth, transcribirlo y mandarlo al servidor.		

Tabla 3.22 RFP-05 Grabado de audio

Grabado de vídeo			**RFP-06**
Dependencias	RFG-05	Prioridad	1
Descripcion	La aplicación debe de ser capaz desde la vista de streaming de grabar el video de la cámara y mandarlo al servidor.		

Tabla 3.23 RFP-06 Grabado de vídeo

Recepción de preguntas			**RFP-07**
Dependencias	RFG-05	Prioridad	2
Descripcion	La aplicación debe de ser capaz desde la vista de streaming de reproducir las preguntas que le lleguen al profesor por dispositivo bluetooth.		

Tabla 3.24 RFP-07 Recepción de preguntas

4. Requisitos del servidor (RFS):

Guardado de discurso			**RFS-01**
Dependencias	RFG-05	Prioridad	2
Descripcion	El servidor debe de guardar automáticamente cada cierto tiempo el discurso completo del profesor para después mostrarlo en el repositorio.		

Tabla 3.25 RFS-01 Guardado de discurso

Guardado de preguntas			**RFS-02**
Dependencias	RFG-05	Prioridad	2
Descripcion	El servidor debe de crear un hilo automáticamente en el foro con las preguntas que haya respondido el profesor en clase.		

Tabla 3.26 RFS-02 Guardado de preguntas

Mensajes de depuración			RFS-03
Dependencias	-	Prioridad	1
Descripcion	El servidor debe mostrar mensajes de depuración durante su ejecución para poder monitorizar el funcionamiento del sistema.		

Tabla 3.27 RFS-03 Mensajes de depuración

Requisitos de restricción

A continuación se incluyen los requisitos de restricción:

Versión del Sistema Operativo			RR-01
Dependencias	-	Prioridad	1
Descripcion	La aplicación será diseñada para ser compatible con versiones del Sistema Operativo Android 4. o superior.		

Tabla 3.28 RR-01 Versión del Sistema Operativo

Compatibilidad Tabletas			RR-02
Dependencias	-	Prioridad	1
Descripcion	La aplicación del alumno debe de ser compatible con tamaños de pantalla de 7" a 10", formato tableta.		

Tabla 3.29 RR-02 Compatibilidad Tabletas

Imposibilidad de salida del Sistema			**RR-03**
Dependencias	-	Prioridad	1
Descripcion	El usuario no podrá salir inesperadamente del sistema sin antes una confirmación.		

Tabla 3.30 RR-03 Imposibilidad de salida del sistema

Permisos de Android			**RR-04**
Dependencias	-	Prioridad	1
Descripcion	Al instalar la aplicación el usuario deberá permitir a la aplicación dar ciertos permisos para su correcto funcionamiento: • Acceso a Internet • Cámara • Bluetooth • Consultar el estado de la red • Leer y escribir en la unidad de almacenamiento • Grabar audio		

Tabla 3.31 RR-04 Permisos de Android

Proteccion de datos			**RR-05**
Dependencias	-	Prioridad	1
Descripcion	No se permitirá el envio de datos confidenciales en claro a través de la red, por lo que las contraseñas estarán cifradas.		

Tabla 3.32 RR-05 Protección de datos

Características del servidor			**RR-06**
Dependencias	-	Prioridad	1
Descripcion	Para el correcto funcionamiento del servidor este debe de cumplir un minimo de condiciones: • Versión mínima de Java 1.8.0_20 • Minimo 500 MB de Ram • Conexión de Internet de banda ancha • Libre el puerto 9000 para las comunicaciones		

Tabla 3.33 RR-06 Características del servidor

Necesidad de equipamiento complementario			**RR-07**
Dependencias	-	Prioridad	2
Descripcion	Para el correcto funcionamiento de todas el sistema es recomendble que junto a los dispositivos Android se disponga de unos auriculares bluetooth y de un tripode para aguantar el dispositvo.		

Tabla 3.34 RR-07 Necesidad de equipamiento complementario

3.2 Diseño

En esta sección se expondrá el diseño de la aplicación. Se analizará el diseño de la arquitectura del sistema, el diseño de la base de datos y el diseño de las vistas de las aplicaciones. Empezaremos con la definición del sistema elegido para desarrollar el proyecto, y terminaremos con la explicación de posibles alternativas de diseño.

3.2.1 Definición del sistema

Una vez hecho el estudio del estado del arte relacionado con la plataforma Android y las tecnologías de servidor, junto con el análisis de los requisitos del sistema se ha hecho un diseño del sistema a desarrollar. El resultado de este proceso ha llevado a decidir que el sistema se compondrá por dos aplicaciones Android y un servidor, que se muestran en la Figura 3.1. Cada aplicación tendrá las siguientes funciones:

(a) BSB App (b) BSB Server (c) Servidor Play

Fig. 3.1 Aplicaciones y servidor del proyecto

1. **BSB App:** Es la aplicación dedicada a los alumnos. Con ella el alumno podrá seguir el streaming de la clase con una transcripción del discurso del profesor, a la vez que toma notas o envía preguntas al profesor. Además cuenta con un repositorio donde se almacena todo el contenido generado durante el streaming y un foro donde poder seguir debatiendo los temas propuestos en clase.

2. **BSB Server:** Es la aplicación del profesor. Él podrá crear sus propias asignaturas donde hará sesiones de streaming de sus clases, todo ello de una forma sencilla e intuitiva. Será capaz de grabar el vídeo y audio, transcribirlo y mandarlo al servidor. También tendrá acceso al foro.

3. **Servidor Play:** Es el servidor que une las comunicaciones y almacena los datos de los usuarios del sistema. Es el encargado de hacer el manejo de las sesiones de streaming y dar servicio a los alumnos que se conecten.

3.2.2 Diseño de la arquitectura

Las conexiones del sistema se basan en una arquitectura Cliente-Servidor. En el proyecto se han desarrollado principalmente tres tipos de comunicaciones diferentes. La primera es la comunicación en HTTP mediante una API REST en la que las aplicaciones Android hacen de cliente realizando peticiones y el servidor Play quien responde a esas peticiones. La segunda es la comunicación mediante WebSockets, que enlaza de una forma sencilla y rápida la aplicación del profesor con varias aplicaciones del alumno a través del servidor. Por último, la conexión Bluetooth entre la aplicación del profesor y el dispositivo Bluetooth de audio.

API REST

La mayoría de peticiones que realizan los clientes Android al servidor están englobados en la API REST desarrollada, como pueden ser pedir el listado de sesiones abiertas, información de perfil, mensajes en el foro o el login.

La comunicación está basada en el patrón de diseño MVC, anteriormente explicado:

- **Modelo:** Representa la información del mundo real que el sistema debe reflejar. En nuestro caso esta representación se realiza mediante el servidor conectándose a la base de datos y reflejando el modelo que representa.

- **Vista:** Es la representación visual del modelo al usuario. Las aplicaciones Android serán nuestra interfaz gráfica del sistema, el cual irá mostrando diferentes partes del modelo según sus peticiones.

- **Controlador:** Recibe y responde a los eventos generados por el usuario interactuando con la vista. El servidor es el encargado de recoger las peticiones de los clientes Android, consultar el modelo en la base de datos, y tras esto generar una respuesta acorde para modificar la interfaz del cliente Android.

Las peticiones HTTP usadas en este proyecto varían entre GET y POST, dependiendo de su función. Todas las peticiones HTTP POST van acompañadas de contenido JSON en el cuerpo de la petición, indicando el contenido que necesita procesar el servidor para generar la respuesta correspondiente, también en formato JSON. En resumen, la comunicación entre cliente y servidor se basa en el intercambio de mensajes en formato JSON que modelan y representan el sistema desarrollado en el proyecto. Como ejemplo de comunicación, se muestran la Figura 3.2 y la Figura 3.3.

```
POST /lisrrooms HTTP/1.1
Host: localhost:9000
Content-Type: application/json
Cache-Control: no-cache
Postman-Token: 041dbdad-cf77-e350-c573-87f312bc9a2f

{"connectionIp":"192.168.43.124","path":"http:\/\/192.168.43.140:9000\/listrooms"}
```

Fig. 3.2 Petición de listado de sesiones abiertas


```
{
  "rooms": [
    {
      "password": null,
      "fotonombre": "p@a.com_Baloncesto.png",
      "fecha_inicio": "2015-08-16 14:30:20.0",
      "email_profesor": "p@a.com",
      "ID_Asignatura": "4",
      "fullname": "Profesor",
      "nombre": "Baloncesto"
    },
    {
      "password": null,
      "fotonombre": "coding_bg.png",
      "fecha_inicio": "2015-08-16 14:29:49.0",
      "email_profesor": "p@a.com",
      "ID_Asignatura": "3",
      "fullname": "Profesor",
      "nombre": "cálculo"
    },
    {...},
    {...},
    {...},
    {...},
    {...},
    {...}
  ]
}
```


Fig. 3.3 Respuesta de listado de sesiones abiertas

WebSockets

La comunicación mediante WebSockets está centrada en la transmisión de audio y vídeo del streaming. Debido a que se trata de una comunicación crítica, que necesita unas características de fiabilidad y rapidez, la comunicación vía sockets es la adecuada.

El diseño de esta comunicación se basa en el diseño de un servidor de chat con salas. En él, la aplicación del profesor es la encargado de crear salas en el servidor, lo que equivalen a sesiones de streaming en el sistema. Tras la creación de las salas, los alumnos pueden entrar a las sesiones de streaming, y por consiguiente a la sala de chat creada. En la Figura 3.4 se muestra un diagrama explicando el esquema de las comunicaciones. Para el intercambio de mensajes en la sala de chat se utiliza el formato JSON, en el que dependiendo de los atributos enviará diferente información y podrá ir dirigido a una sola persona de la sala de chat o a todas:

- **Join:** Indica que una persona ha entrado a la sala de chat. Es notificado a todos las personas dentro de la sala.
- **Stream:** Indica que este mensaje incluye información de audio y vídeo. Este mensaje sólo lo origina la aplicación del profesor y es enviado al resto de la sala.
- **Notification:** Este mensaje es generado por la aplicación del alumno y va dirigida sólo al profesor. Este mensaje indica que un alumno ha realizado una pregunta al profesor por el sistema.
- **Quit:** Indica que una persona ha salido a la sala de chat. Es notificado a todas las personas dentro de la sala.

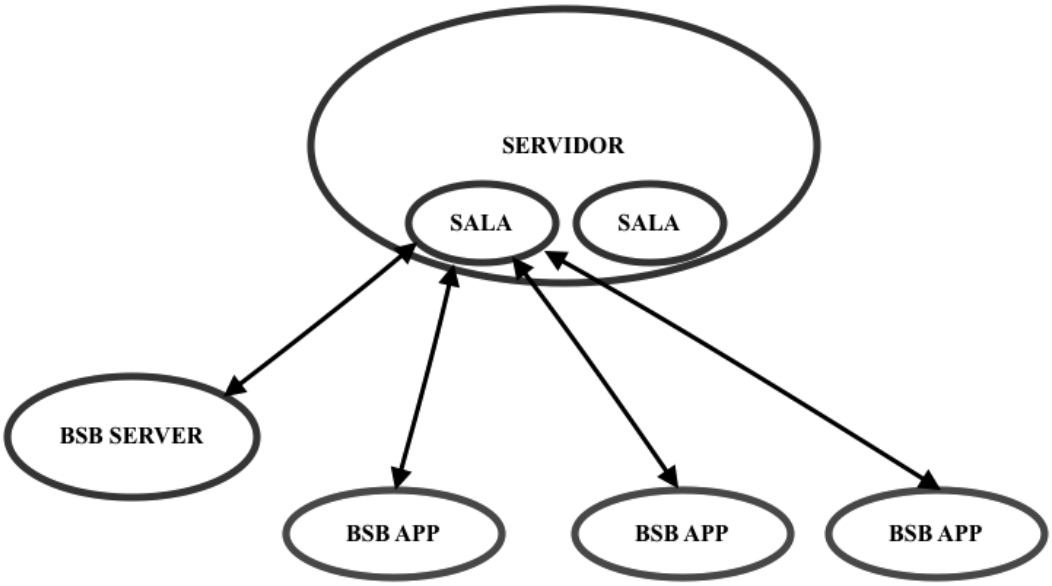


Fig. 3.4 Diagrama de la comunicación vía WebSockets

Bluetooth

La comunicación Bluetooth en este proyecto es utilizada para extender el rango de comunicación del audio al profesor y poder grabar su voz lejos del dispositivo Android. La presencia de esta tecnología en la plena mayoría de dispositivos móviles Android, hace que su uso en este proyecto de unas ventajas de movilidad enormes, necesarias para el uso del sistema. Además, Android maneja las conexiones con estos dispositivos, lo que hace más fácil su implementación.

Bluetooth [33] es un estándar de tecnología inalámbrica para el intercambio de información en distancias cortas. El estándar Bluetooth fue creado por el IEEE como el estándar 802.15.1, el cual pasó a desarrollo de la organización Bluetooth SIG, compuesta por múltiples industrias y compañías que desarrollan y mantienen esta tecnología. Bluetooth opera en la banda espectral sin licenciar industrial, científica y médica de los 2.4 GHz, desde 2.4GHz hasta 2.468 GHZ, mediante la tecnología frequency hopping en intervalos de 1 MHz, para evitar la interferencia entre canales. El alcance normal en los dispositivos móviles es de alrededor de 10 metros, comprendiendo la clase 2 de radio Bluetooth. La comunicación entre dispositivos Bluetooth se basa en una estructura master-slave, en la que un dispositivo

Bluetooth puede comunicarse con hasta otros 7. Todos ellos comparten el reloj de transmisión del dispositivo master. A partir de la designación de roles, cualquiera puede transmitir datos por los ranuras de tiempo disponibles, utilizando el master las ranuras pares y los slave las impares.

Desde la aplicación del profesor detectamos si hay dispositivos Bluetooth conectados, y redirigimos micrófono y altavoz del dispositivo Android al dispositivo Bluetooth. Con ello conseguimos que el profesor pueda alejarse del dispositivo Android, el cual dejará grabando el vídeo de la clase. Así el profesor podrá desplazarse por la clase sin interrumpir las funciones de transcripción de audio y la recepción de preguntas llevadas acabo en el dispositivo Android. En la Figura 3.5 podemos observar un diagrama de las comunicaciones Bluetooth entre la aplicación del profesor y el dispositivo Bluetooth.

Fig. 3.5 Diagrama de la comunicación vía Bluetooth

3.2.3 Diseño de la base de datos

Para el desarrollo del proyecto ha sido necesario almacenar cierto tipo de información, necesaria tanto para la gestión de datos de los usuarios, como para el propio funcionamiento del sistema. Para facilitar esa tarea en el servidor, se utilizará el gestor de bases de datos MySQLl.

Los usuarios no son los que hacen directamente la petición a la base de datos. Estos mandan una petición al servidor y este les responde con la información requerida, siguiendo el modelo MVC anteriormente explicado. Por lo que es el servidor el que hace consultas en el lenguaje SQL al gestor de base de datos.

En la Figura 3.6 se detalla el modelo Entidad-Relación [34], el cual modela la información de la base de datos y sus relaciones, al tratarse de una base de datos relacional. Cada cuadro representa una tabla de la base de datos. Dentro de cada cuadro están definidos los atributos, con su tipo de datos y si son clave primaria (llave) o foránea (punto rojo). Las flechas indican el tipo de relación entre tablas.

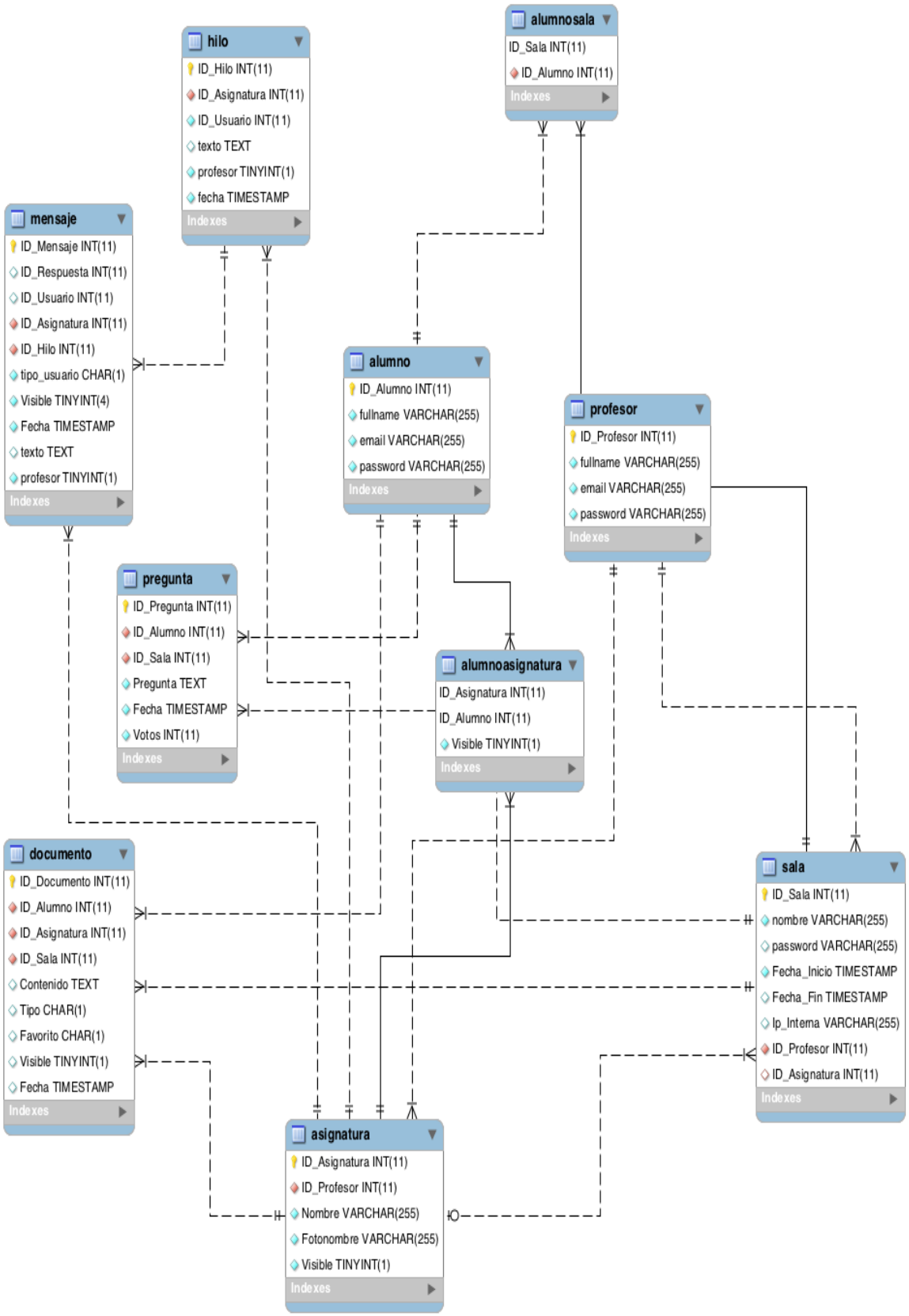


Fig. 3.6 Diagrama de la base de datos

A continuación se explicará el porqué de cada tabla y sus atributos:

1. **Alumno:** Contiene la información de usuario de los alumnos registrados en el sistema. Los alumnos están separados de los profesores en diferentes tablas para el correcto funcionamiento del modelo del sistema ya que las opciones de uno y otro son completamente diferentes.

    - **ID_Alumno:** Identifica inequívocamente a un alumno dentro del sistema.
    - **Fullname:** Nombre completo del alumno.
    - **Email:** Email de acceso a la aplicación.
    - **Password:** Contraseña de acceso a la aplicación.

2. **AlumnoAsignatura:** Contiene la información que relaciona cada alumno con cada asignatura a la que ha accedido.

    - **ID_Asignatura:** Referencia la asignatura.
    - **ID_Alumno:** Referencia al alumno.
    - **Visible:** Este campo se usa para mostrar la lista de asignaturas del alumno si este no las ha borrado del historial.

3. **AlumnoSala:** Contiene la información que relaciona cada alumno con cada sesión a la que ha acudido para poder referenciar los contenidos del repositorio por sesión acudida.

    - **ID_Sala:** Referencia la asignatura.
    - **ID_Alumno:** Referencia al alumno.

4. **Asignatura:** Contiene la información de las asignaturas que imparten los profesores.

    - **ID_Asignatura:** Identifica inequívocamente a una asignatura dentro del sistema.
    - **ID_Profesor:** Referencia al profesor que imparte la asignatura.
    - **Nombre:** Nombre completo de la asignatura.
    - **FotoNombre:** Nombre del archivo almacenado en los recursos del servidor para la imagen de fondo correspondiente a la asignatura.
    - **Visible:** Este campo se usa para mostrar la lista de asignaturas del profesor si este no las ha borrado del historial.

5. **Documento:** Contiene la información de documentos del sistema, como pueden ser capturas de pantalla o texto de apuntes.

    - **ID_Documento:** Identifica inequívocamente a un documento dentro del sistema.
    - **ID_Alumno:** Referencia al alumno que pertenece este documento.
    - **ID_Asignatura:** Referencia a la asignatura que pertenece este documento.
    - **ID_Sala:** Referencia a la sesión en la que se creó este documento.
    - **Contenido:** Puede contener el nombre de la captura de pantalla o el contenido de texto de un apunte tomado, depende del atributo Tipo.
    - **Tipo:** Este campo se usa para determinar si el documento es una captura de pantalla o un apunte tomado.
    - **Favorito:** Indica si el documento es favorito para el alumno.
    - **Visible:** Este campo se usa para mostrar el documento si este no lo ha borrado del historial.
    - **Fecha:** Indica la fecha de toma del documento.

6. **Hilo:** Referencia a un hilo de conversación dentro de una asignatura.

    - **ID_Hilo:** Identifica inequívocamente a un hilo dentro del sistema.
    - **ID_Asignatura:** Referencia a la asignatura que pertenece este hilo.
    - **ID_Usuario:** Referencia al usuario que pertenece este hilo, dependiendo del atributo profesor será un alumno o profesor.
    - **Texto:** Indica el título del hilo.
    - **Profesor:** Este campo se usa para determinar si el hilo fue creado por un profesor o alumno y poder referenciarlo bien con su id.
    - **Fecha:** Indica la fecha de creación del hilo.

7. **Mensaje:** Referencia a un mensaje de texto dentro de un hilo.

    - **ID_Mensaje:** Identifica inequívocamente a un mensaje dentro del sistema.
    - **ID_Respuesta:** Identifica el ID_Mensaje al que responde, si el mensaje es una contestación a otro mensaje.
    - **ID_Usuario:** Referencia al usuario que pertenece este mensaje, dependiendo del atributo profesor será un alumno o profesor.

- **ID_Asignatura:** Referencia a la asignatura que pertenece este mensaje.
- **ID_Hilo:** Referencia al hilo que pertenece este mensaje.
- **Visible:** Este campo se usa para mostrar el mensaje si este no lo ha borrado el usuario.
- **Fecha:** Indica la fecha de creación del mensaje.
- **Texto:** Indica el contenido de texto del mensaje.
- **Profesor:** Este campo se usa para determinar si el mensaje fue creado por un profesor o alumno y poder referenciarlo bien con su id.

8. **Pregunta:** Contiene las preguntas realizadas por los alumnos durante una sesión de streaming.

   - **ID_Pregunta:** Identifica inequívocamente a una pregunta dentro del sistema.
   - **ID_Alumno:** Referencia al alumno que pertenece esta pregunta.
   - **ID_Sala:** Referencia a la sesión en la que se creó esta pregunta.
   - **Pregunta:** Indica el contenido de texto de la pregunta.
   - **Fecha:** Indica la fecha que se realizó la pregunta.
   - **Votos:** Este campo se usa para llevar la cuenta de votos que ha recibido la pregunta.

9. **Profesor:** Contiene la información de usuario de los profesores registrados en el sistema.

   - **ID_Profesor:** Identifica inequívocamente a un profesor dentro del sistema.
   - **Fullname:** Nombre completo del profesor.
   - **Email:** Email de acceso a la aplicación.
   - **Password:** Contraseña de acceso a la aplicación.

10. **Sala:** Contiene la información de las sesiones creadas dentro del sistema.

    - **ID_Sala:** Identifica inequívocamente a una sala dentro del sistema.
    - **Nombre:** Nombre del profesor que creó la sala.
    - **Password:** Contraseña de acceso a la sesión.
    - **Fecha_Inicio:** Indica la fecha que se inició la sesión.
    - **Fecha_Fin:** Indica la fecha que se finalizó la sesión.

- **Ip_Interna:** Indica la IP desde donde se inició la sesión.
- **ID_Profesor:** Referencia a un profesor dentro del sistema.
- **ID_Asignatura:** Referencia a la asignatura que pertenece este hilo.

3.2.4 Diseño de las vistas de las aplicaciones

En este apartado definiremos las interfaces de las aplicaciones del proyecto. Cada interfaz ha sido diseñada pensando en el usuario final dependiendo si es para un profesor o alumno. Ambas aplicaciones han seguido las guías de diseño de la Fundación Adecco, entre las que se encuentran tipografías, fotos corporativas, iconos y colores.

La aplicación del profesor se ha intentado crear una interfaz lo más sencilla e intuitiva posible. Se supone un perfil de usuario no muy experto en tecnologías móviles, que además necesita perder el mínimo de tiempo posible en configuraciones para poder dar la clase. Por ello, la vista de configuración de asignaturas y de streaming son interfaces simples y fáciles de manejar.

La aplicación del alumno se ha diseñado de una forma que pueda ofrecer muchas más características, sin llegar a ser difícil de manejar. Se supone un usuario joven, acostumbrado a tratar con tecnologías en el día a día y que necesita sacar el máximo provecho de las aplicaciones que utiliza. Por ello, la vista de streaming es mucho más completa que la del profesor, permitiendo al alumno tomar notas, hacer preguntas y tomar capturas de pantalla mientras está recibiendo el streaming.

A continuación se explica en detalle los diseños de las vistas de las dos aplicaciones:

Módulo login

La Figura 3.7 muestra la pantalla que aparece tras entrar en cualquiera de las dos aplicaciones. En ella el usuario deberá introducir sus credenciales para acceder al sistema. Si el email no es válido o la contraseña no es correcta, habrá una notificación visual para alertar al usuario. Además existe la opción de recordar contraseña, para que la próxima vez que se entre en la aplicación en login se realice de forma automática.

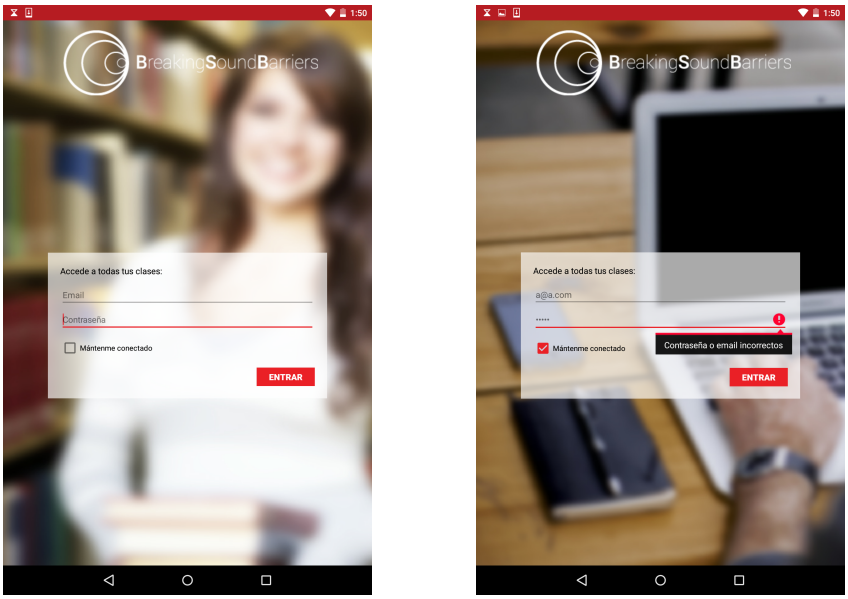


Fig. 3.7 Vista de Login

Módulo vista principal

- **Aplicación profesor:**

  Nada más pasar el login, el profesor aparecerá en esta vista, donde se muestran las asignaturas que ha creado. Pulsando en ellas accederá directamente a la configuración de una nueva sesión para empezar el streaming. Aquí podrá configurar las características de la sesión, y elegir entre si la sesión es privada o requiere contraseña.

  Si el profesor no tiene todavía ninguna asignatura configurada, podrá crear una nueva siguiendo unos pasos sencillos en los que definirá el nombre y la foto de la asignatura a personalizar. Con esto se pretende que el profesor tenga un acceso rápido y fácil al inicio de sesiones de su asignatura y a la configuración de nuevas asignaturas. La Figura 3.7 muestra la pantalla de la vista principal del profesor.

- **Aplicación alumno:**

  Por otro lado, la vista principal del alumno difiere bastante de la del profesor. Esta vista pretende ser más una vista de inicio que muestre los últimos eventos que han surgido en la aplicación. Aquí aparecerán las ultimas capturas de pantalla y apuntes que se hayan tomado. Además se incluyen unas tarjetas de ayuda que explican el funcionamiento de todas las secciones de la aplicación. La Figura 3.9 muestra la pantalla de la vista principal del alumno.

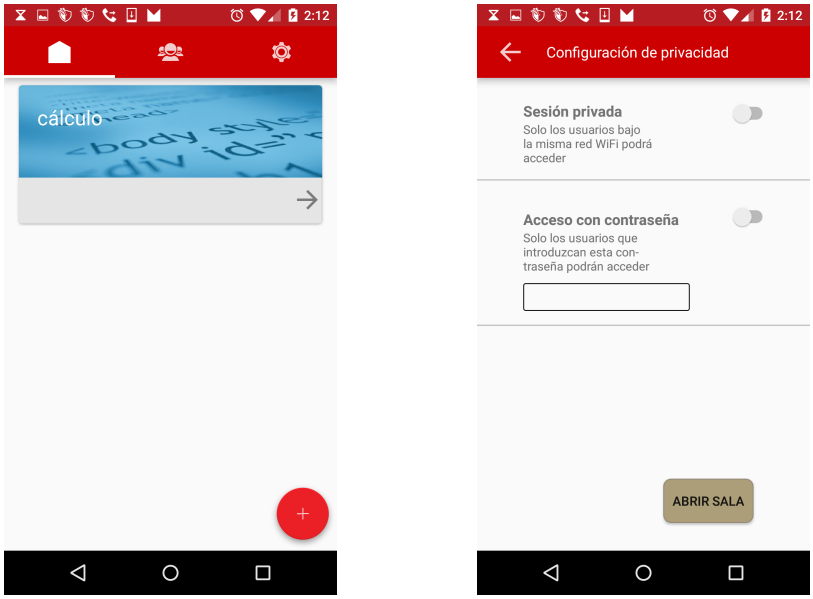


Fig. 3.8 Vista principal profesor

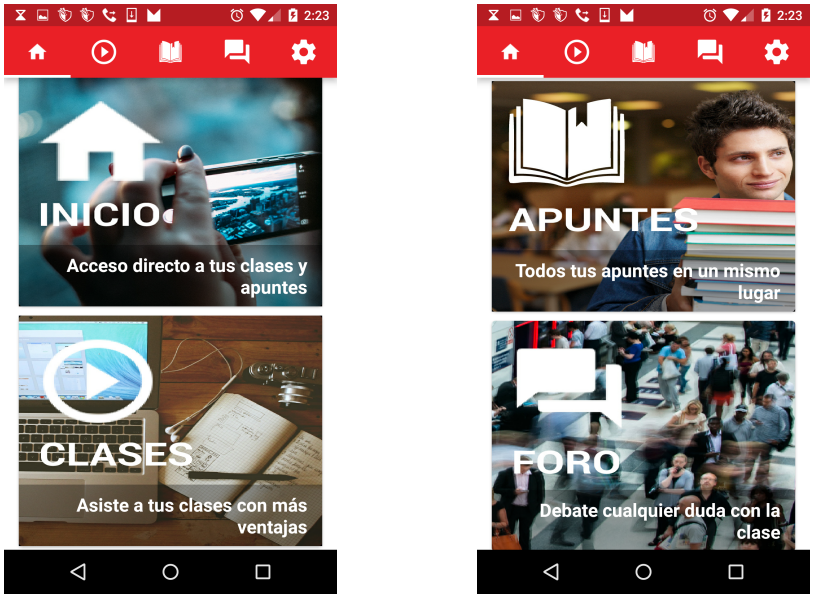


Fig. 3.9 Vista principal alumno

Módulo streaming

- **Aplicación profesor:**

  Una vez seleccionada la asignatura que quiere abrir sesión el profesor, entrará en la vista de streaming. Aquí se muestra la cámara y lo que se va a transmitir cuando se empiece a grabar. Tras dar al botón de play, el streaming de vídeo y audio empezará a transmitirse.

  Para finalizar la sesión el profesor podrá dar al botón de stop, que aparece después de dar al play, o dar al botón de atrás y salir de la vista de streaming. Desde esta vista también se avisa al profesor si no se detecta que hay conectado un dispositivo Bluetooth al dispositivo Android, por lo que no se podrá traducir ni el audio, ni las preguntas que lleguen al profesor. La Figura 3.10 muestra la pantalla de la vista de streaming del profesor.

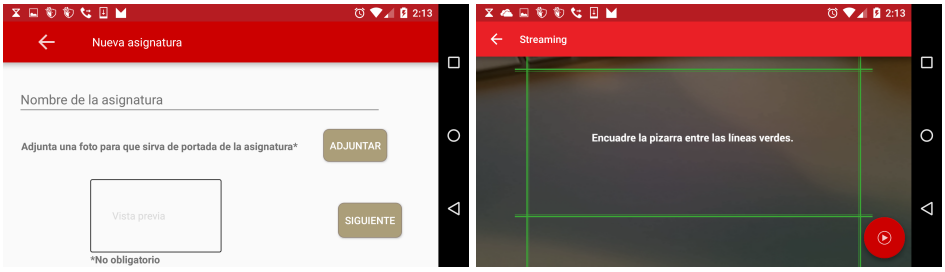


Fig. 3.10 Vista streaming profesor

- **Aplicación alumno:**

  El alumno podrá entrar a la vista de streaming dando al segundo botón de la barra de navegación de la aplicación. Allí se mostrarán las sesiones abiertas, a las que podrá acceder si cumple los requisitos de privacidad marcados por el profesor de la signatura para esa sesión. Además se facilita una función de búsqueda para filtrar las sesiones por nombre de profesor o asignatura. Una vez seleccionado entrará en una sesión.

  La vista de streaming dentro de la sesión se divide en dos secciones principales. La primera, a la izquierda, se muestra el vídeo y el audio transcrito. La segunda, a la derecha, contiene una serie de opciones que complementan al streaming. Desde ahí, el alumno puede tomar notas o capturas de pantalla, que después aparecerán en el repositorio, además de formular preguntas al profesor o votar las ya formuladas por otros compañeros en la misma sesión. La Figura 3.11 muestra la pantalla de la vista de streaming del alumno.

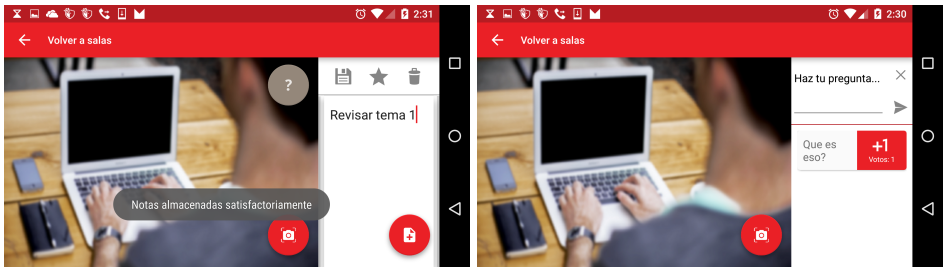


Fig. 3.11 Vista streaming alumno

Módulo foro

Una vez en la vista del foro, aparecerán para participar en el foro aquellas asignaturas en las que hayas presenciado una sesión de ellas. Una vez dentro del foro de la asignatura, se mostrarán los hilos actualmente abiertos, con posibilidad de crear nuevos. Dentro de cada hilo aparecerán los mensajes de la conversación de ese hilo. La Figura 3.12 muestra la pantalla de la vista de foro.

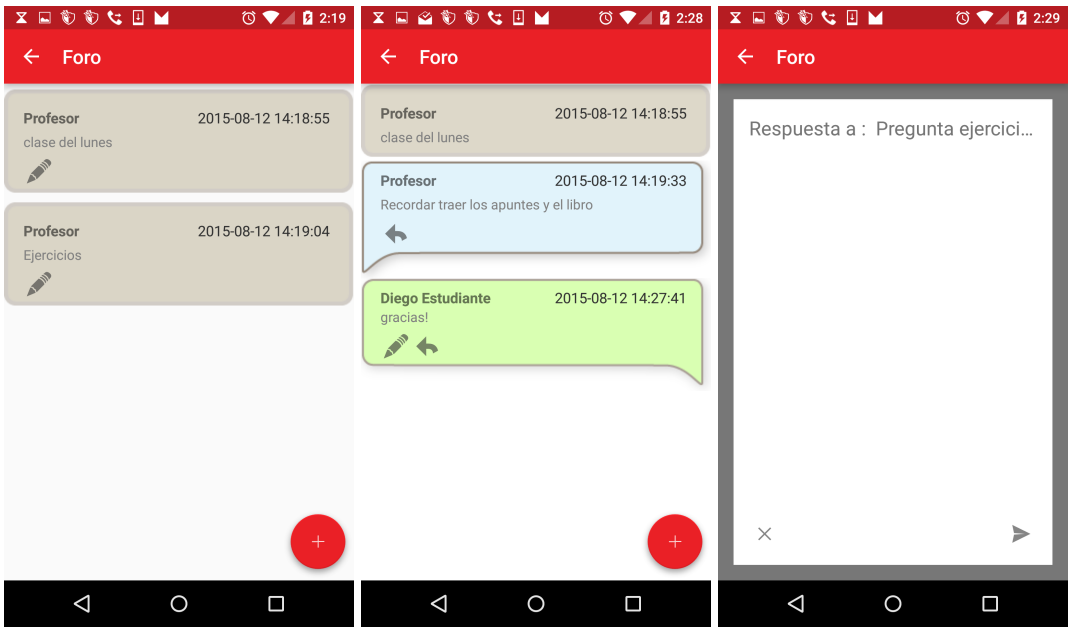


Fig. 3.12 Vista foro

Módulo repositorio

Al igual que la vista de foro, al entrar en ella se mostrarán las asignaturas a las que hayas asistido en alguna sesión. Aquí aparecerán las notas y capturas de pantalla que hayas

realizado durante una sesión de streaming. Además, aunque no se hayan tomado apuntes ni realizado capturas, aparecerá el discurso del profesor transcrito durante esa sesión. Los documentos en el repositorio se almacenan cronológicamente y por sesión atendida. La Figura 3.13 muestra la pantalla de la vista de streaming.

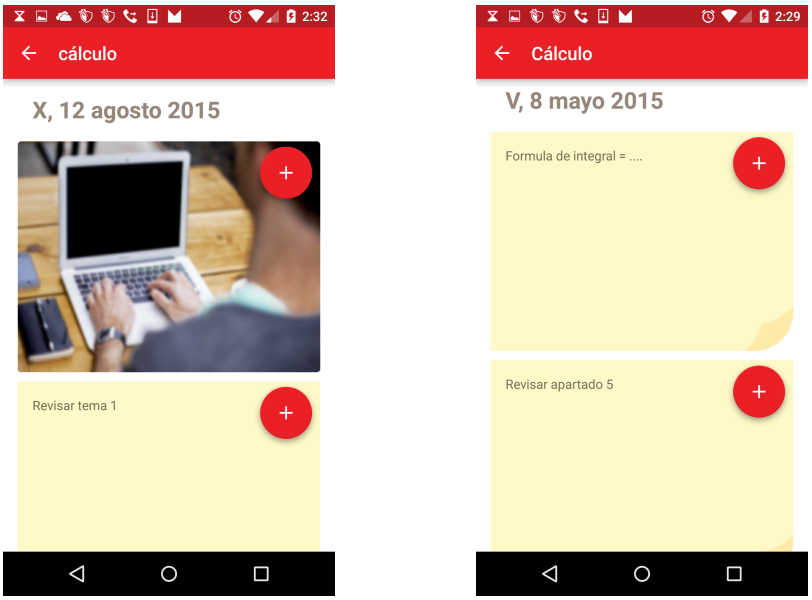


Fig. 3.13 Vista repositorio

Módulo vista usuario

Esta es la vista de configuración del perfil de usuario, parecidas en ambas aplicaciones. Tanto profesores como alumnos podrán modificar su nombre, avatar y contraseña. Por el lado del profesor, este además podrá editar sus asignaturas, cambiar de foto, nombre e incluso borrarlas. Mientras que por el lado del alumno, este podrá configurar ajustes como el tamaño de los subtítulos en el streaming, borrar su historial de asignaturas o borrar el repositorio. La Figura 3.14 muestra la pantalla de la vista de usuario.

Otros elementos de diseño a parte de los mencionados en cada módulo se muestran en la Figura 3.15. La descripción a continuación:

- La pantalla durante el streaming se mantiene siempre encendida y no se bloquea, tanto para la aplicación del alumno como para la del profesor, así evitamos estar pendientes de mantener la pantalla encendida manualmente y evitamos los bloqueos automáticos.

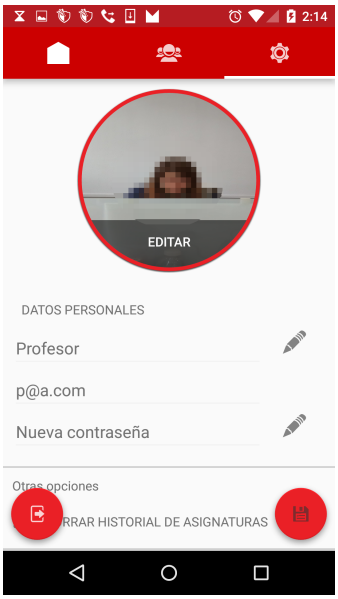


Fig. 3.14 Vista usuario

- Durante las peticiones al servidor se muestra una animación de carga, para avisar al usuario que la aplicación esta trabajando en mostrar los recursos.
- Los iconos de la aplicación intentan ser auto-explicativos de su función, para ello se han escogido los iconos que recomienda Google para el diseño de aplicaciones.

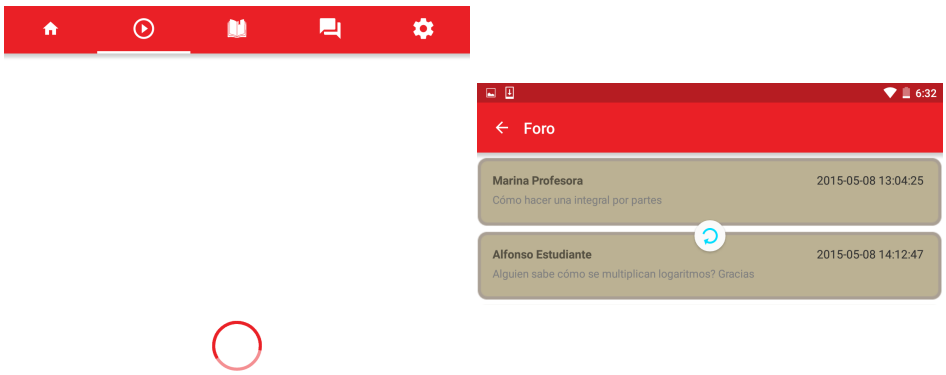


Fig. 3.15 Animaciones de progreso progress bar y pull refresh

3.2.5 Alternativas

A continuación se comentan brevemente las diversas opciones que se plantearon en el momento de diseñar el sistema y se explica el motivo por el que fueron descartadas.

Selección de plataforma móvil

El primer motivo de discusión fue la elección de la plataforma móvil, donde desarrollar las aplicaciones del alumno y el profesor. Entre ellas se ofrecían como alternativas los sistemas operativos Windows Phone e iOS. Windows Phone fue la primera en ser descartada debido a la poca popularidad de la plataforma en dispositivos móviles y sobre todo en tabletas. Por otro lado, iOS cuenta con una gran mercado de usuarios y una gran plataforma de desarrollo. Los principales motivos por los que se descartó en favor a Android son los siguientes:

- Para desarrollar en iOS es necesario contar con una subscripción de desarrollador anual, dispositivos iOS para poder probar las aplicaciones y ordenadores Mac para poder usar el entorno de desarrollo. El equipo de desarrolladores de este proyecto contaba con un sólo Mac y ninguna de las otras condiciones.

- El precio de los dispositivos es considerablemente elevado en comparación con las diversas opciones de precio de los dispositivos Android. Pensando en un posible despliegue en una Universidad, por el precio de un móvil con sistema operativo iOS, se podrían utilizar al menos dos tabletas para los alumnos y un móvil para el profesor.

- Total inexperiencia del equipo para desarrollar en la plataforma iOS con el lenguaje de programación Swift. Mientras que Android está basado en Java, del cual se tenían suficientes conocimientos previos.

Formato de mensajes

A la hora del intercambio de mensajes entre el cliente y servidor las tecnologías predominantes son XML y JSON, contando con un formato leíble a la concepción humana. XML cuenta con gran popularidad en el desarrollo de aplicaciones web para el intercambio de información. Además, se contaba con experiencia en el desarrollo de este tipo de mensajes. Aún así, se optó por utilizar el formato JSON, por la versatilidad que ofrece a la hora del proceso y su menor tamaño debido al menor uso de etiquetas para el formateo de la información. También ayudó a la decisión el soporte nativo de Android para el parseo de objetos JSON.

Tipo de base de datos

El sistema desarrollado necesita de una gran cantidad de datos que almacenar, por lo que la decisión del tipo de base de datos fue crítico para la implementación del proyecto. La principal alternativa fue el uso de la base de datos documental MongoDB, que ofrece una gran rapidez para el proceso de peticiones cuando hay muchos datos almacenados, además de la flexibilidad de modificación de campos en las tablas. Por otro lado, la utilización de este tipo de base de datos no permitía establecer el modelo del sistema al no poder relacionar las tablas, y esto debía de implementarse en la parte del servidor. Por ello, se decidió la utilización de una base de datos relacional MySQL, la cual permitía seguir el modelo del sistema y un tiempo de respuesta de proceso aceptable, al no tener que almacenar una cantidad de datos para que resultara positivo el cambio.

Comunicaciones por sockets

Para la comunicación del vídeo y audio desarrollada en el proyecto, se necesitaba un tipo de comunicación en el que se estableciese un canal fiable e ininterrumpido de transmisión de información, un socket. La principal alternativa que se planteó fue la utilización de una comunicación por sockets TCP directa entre las aplicaciones Android del profesor y los alumnos. Debido a la utilización de un servidor web con soporte para WebSockets, se proponía más fácil la adaptación de las aplicaciones Android a la infraestructura de comunicaciones del servidor, que construir de cero una comunicación TCP entre los dispositivos.

3.3 Implementación

A continuación se describe el proceso de implementación del sistema desarrollado en este proyecto. La implementación ha sido llevada a cabo mediante módulos independientes atendiendo a las principales funciones de las aplicaciones móviles Android.

3.3.1 Entorno tecnológico

En esta sección se explican las necesidades tecnológicas utilizadas para la implementación del sistema descrito en el proyecto.

Equipo

El equipo utilizado para la implementación y prueba del sistema es el siguiente:

- **Ordenador portátil MacBook Air** 2012 13", 8 GB de RAM, disco duro flash 128 GB y procesador Intel i5 a 1.8 GHz. Cuenta con el JDK de Java versión 1.8.0_20, el SDK de Android, Play Framework, MySQL y Git instalados.
- **Dispositivo Bluetooth BeeWi BBH100**, cascos inalámbricos con capacidad de reproducción de música y recepción de llamadas.
- **Dipositivo móvil Google Nexus 5**, probado desde la versión 4.4.4 a la 5.2 de Android.
- **Dispositivo tableta Google Nexus 7**, probado desde la versión 4.3.0 a la 5.1.1 de Android.

Programas

Los programas utilizados para el desarrollo del proyecto son los siguientes:

- **Android Studio:** Entorno de desarrollo para las aplicaciones Android.
- **Intellij IDEA 14 CE:** Entorno de desarrollo para el desarrollo del servidor.
- **MySQL Workbench:** Programa para la gestión de la base de datos a través de una interfaz gráfica.
- **Adobe Photoshop e Illustrator:** Programas de la suite de Adobe para el retoque fotográfico y diseño de las aplicaciones Android.
- **SourceTree:** Programa para la gestión del control de versiones de Git con interfaz gráfica.

Para el desarrollo de la memoria:

- **TexMaker:** Entorno de desarrollo para creación de documentos en el lenguaje LaTex, un sistema de composición de textos, orientado a la creación de documentos escritos.

- **Overleaf.com:** Herramienta web para la edición de documentos LaTex.
- **Smartsheet.com:** Herramienta web para la planificación y desarrollo de proyectos.

3.3.2 Módulo comunicaciones

El módulo de comunicaciones comprende la implementación de las comunicaciones de la API REST sobre HTTP, la conexión WebSocket entre las aplicaciones Android y el servidor, la comunicación con el dispositivo Bluetooth y la conexión del servidor con la base de datos.

API REST

Como anteriormente se comentó, el diseño de la API se basa en el patrón de diseño MVC. La parte del controlador estará implementada en el servidor, el modelo vendrá dado por la base de datos y la vista serán las aplicaciones Android. La comunicación entre el servidor y las aplicaciones se hará mediante el protocolo HTTP.

En la parte del servidor tendremos dos clases que controlarán el manejo de las peticiones dependiendo del recurso que soliciten en el servidor. Estas son *Application* y *Routes* mostradas en la Figura 3.16 y en la Figura 3.17.

Fig. 3.16 Clase *Application* para la comunicación vía HTTP del servidor

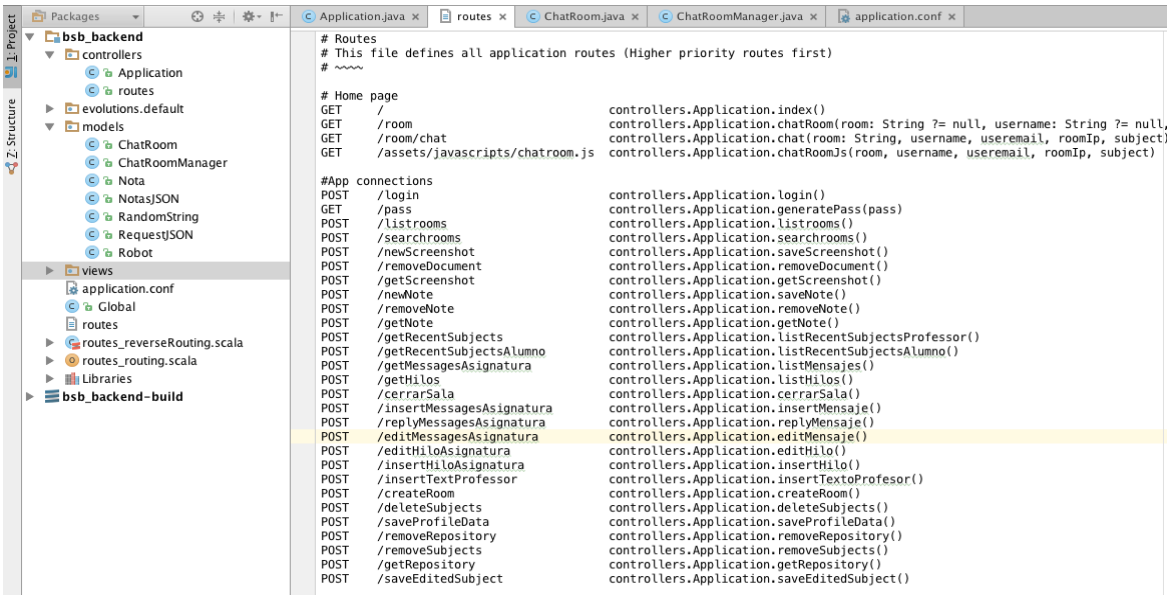


Fig. 3.17 Clase *Routes* para la comunicación vía HTTP del servidor

Las rutas expuestas en la clase *Routes* son las utilizadas como dirección en las peticiones realizadas por las aplicaciones Android. Estas peticiones se realizan mediante las clases *TareaAsincrona* y *RespuestaAsincrona*, mostradas en la Figura 3.18 y en la Figura 3.18, las cuales utilizan la clase de Android *AsyncTask* y *HttpClient* para realizar las peticiones en segundo plano y poder esperar a la respuesta sin bloquear la vista del usuario en la aplicación.

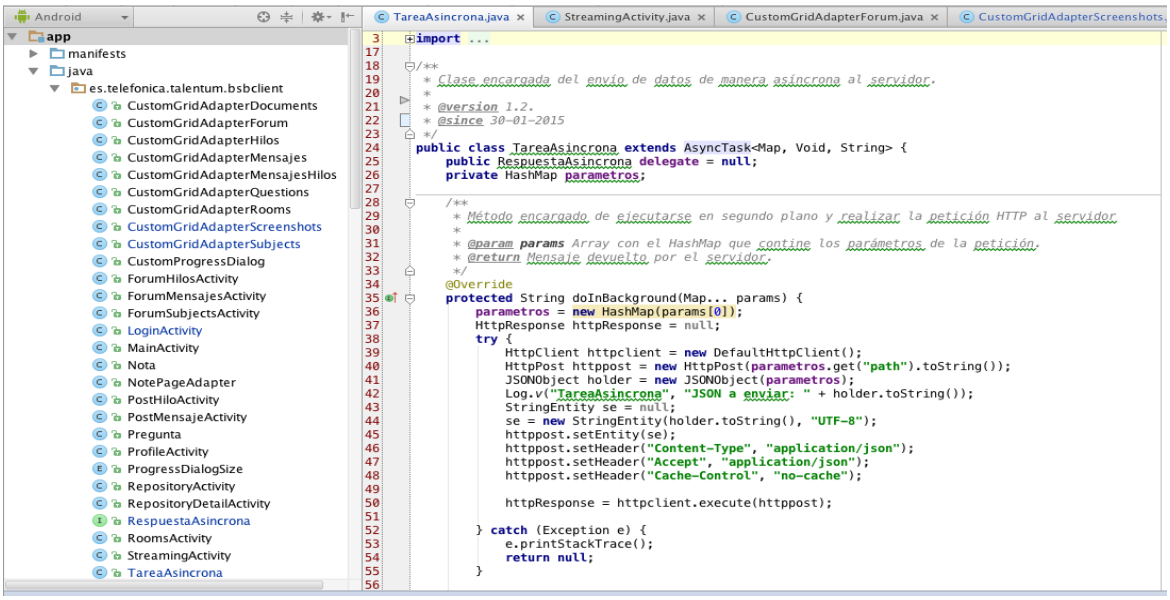


Fig. 3.18 Clase *TareaAsincrona* para la comunicación vía HTTP de las aplicaciones Android

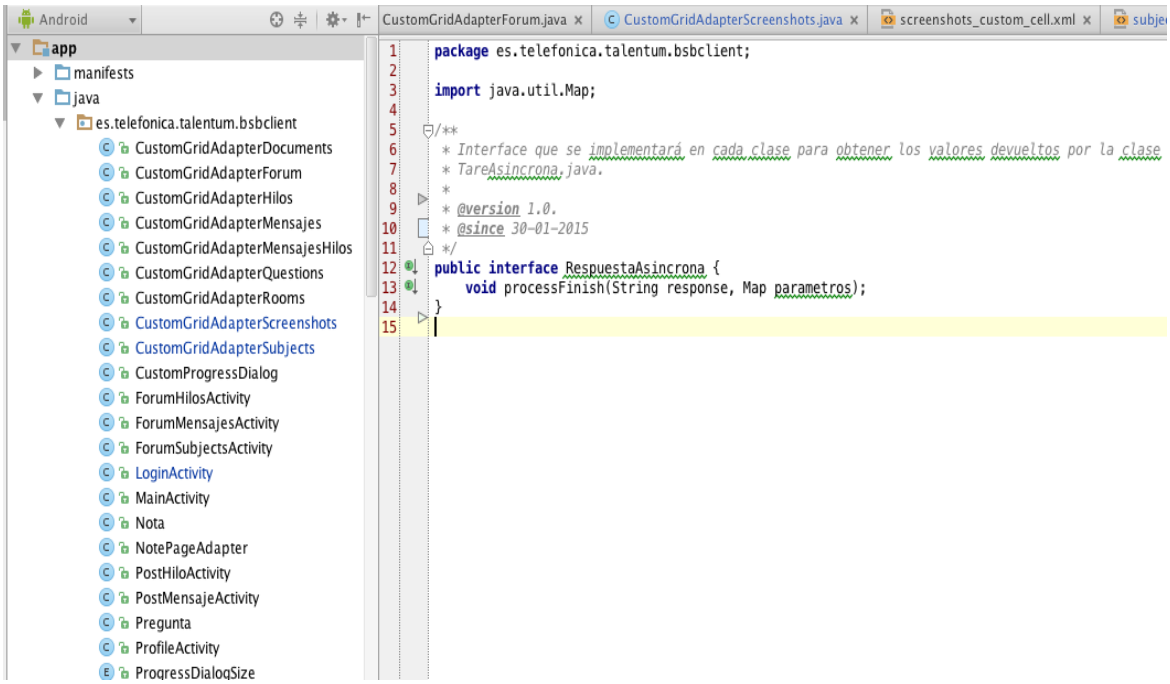


Fig. 3.19 Clase *RespuestaAsincrona* para la comunicación vía HTTP de las aplicaciones Android

Conexión WebSocket

La conexión vía WebSockets está implementada en el servidor siguiendo la estructura de un servidor de chat, en el que puede haber varias salas con diferentes personas hablando en ellas. Esto se asemeja a nuestro modelo de comunicación en las sesiones de streaming. En nuestro caso, es el profesor el que inicia en el servidor una sala de chat vía WebSockets y los alumnos se conectan a ella. Cuando el profesor habla, es decir, transmite vídeo y audio del streaming, todos los alumnos que estén conectados en la misma sala lo recibirán. Lo mismo ocurre con las preguntas en el sentido contrario. Para ello en el servidor hay dos clases encargadas de esta gestión, *ChatRoom* y *ChatRoomManager*, mostradas en la Figura 3.20 y en la Figura 3.21.

```
package models;

import ...

/**...*/
public class ChatRoomManager extends ChatRoom {

    // Default room.
    //static ActorRef defaultRoom = Akka.system().actorOf(Props.create(ChatRoom.class));
    static Map<Integer, ActorRef> rooms = new HashMap<>();

    /**
     * Join the default room.
     */
    public static void join(String roomName, final String username, final String useremail, final String
        final ActorRef room;
        Logger.debug("Join executed");
        if (rooms.containsKey(roomName.hashCode())){
            room = rooms.get(roomName.hashCode());
        } else {
            room = Akka.system().actorOf(Props.create(ChatRoomManager.class));
            rooms.put(roomName.hashCode(),room);
            Logger.debug("Sala creada: "+roomName);
        }

        // Send the Join message to the room
        String result = (String)Await.result(ask(room,new Join(username, out, useremail, roomIp, subject)

        if("OK".equals(result)) {

            // For each event received on the socket,
            in.onMessage(new Callback<JsonNode>() {
               public void invoke(JsonNode event) {
                    if(event.get("notification").asText().equals("true")){
                        room.tell(new Notification(username, event.get("text").asText(), useremail),null);
                    }
                    else if(event.get("stream").asText().equals("true")){
                        room.tell(new Stream(username, event.get("text").asText()),null);

                    }
                    else if(event.get("forceQuit").asText().equals("true")){
                        room.tell(new ForceQuit(username),null);
                    }
                    else {
                        room.tell(new Talk(username, event.get("text").asText(),useremail), null);

                    }
               }
            });
```

Fig. 3.20 Clase *ChatRoomManager* para la comunicación vía WebSockets en el servidor

```
ChatRoom.java | ChatRoomManager.java | Application.java | routes | application.conf

package models;

import ...

/**...*/
public class ChatRoom extends UntypedActor {
    // Members of this room.
    Map<String,WebSocket.Out<JsonNode>> members = new HashMap<>();
    String profesor = null;
    public void onReceive(Object message) throws Exception {...}

    // Send a Json event just to a member
    public void notify(String kind, String user, String text) {
        //WebSocket.Out<JsonNode> channel = members.get(user);
        WebSocket.Out<JsonNode> channel = members.get(profesor);

        ObjectNode event = Json.newObject();
        event.put("kind", kind);
        event.put("user", user);
        event.put("message", text);

        ArrayNode m = event.putArray("members");
        for(String u: members.keySet()) {
            m.add(u);
        }

        Logger.debug("Kind: "+kind+". User: "+user+". Message: "+text);
        channel.write(event);

    }

    // Send a Json event to all members
    public void notifyAll(String kind, String user, String text) {
        for(WebSocket.Out<JsonNode> channel: members.values()) {

            ObjectNode event = Json.newObject();
            event.put("kind", kind);
            event.put("user", user);
            event.put("message", text);

            ArrayNode m = event.putArray("members");
            for(String u: members.keySet()) {
                m.add(u);
            }

            channel.write(event);
        }
    }

    public static class Join {
```

Fig. 3.21 Clase *ChatRoom* para la comunicación vía WebSockets en el servidor

Por parte de las aplicaciones Android existen varios métodos implementados, mostrados en la Figura 3.22 y en la Figura 3.23, para el manejo de los mensajes mediante el canal WebSocket. Para ello se utiliza la clase de Android *WebSocketClient* que nos permite manejar la recepción y envío de mensajes a través de WebSockets, además de la inicialización de la comunicación.

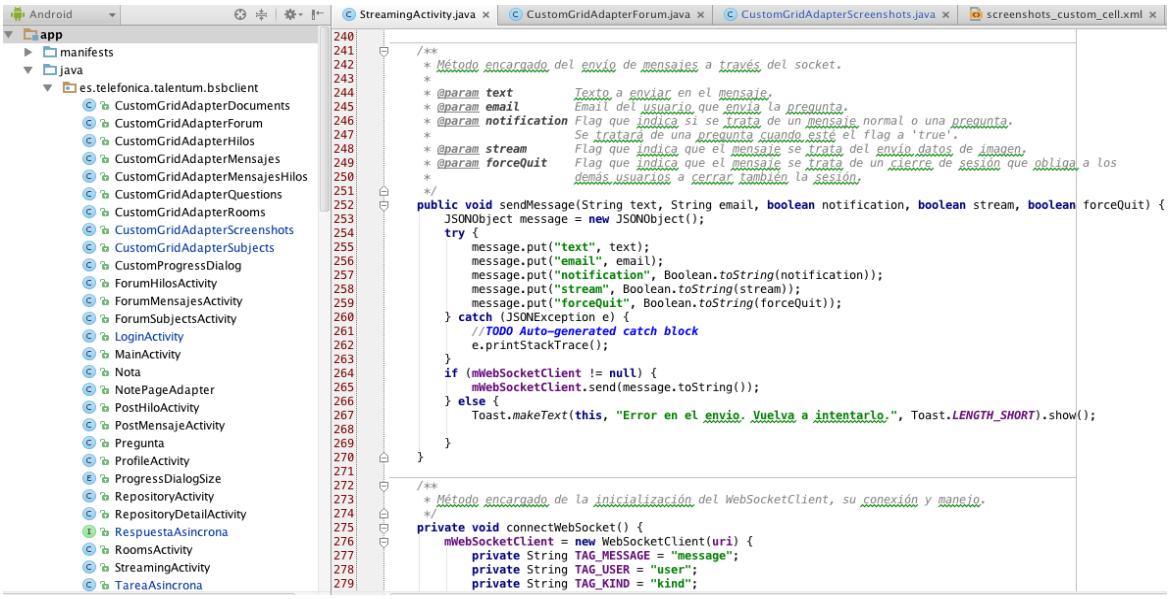


Fig. 3.22 Método *sendMessage* para el envío de mensajes vía WebSockets en la aplicación Android

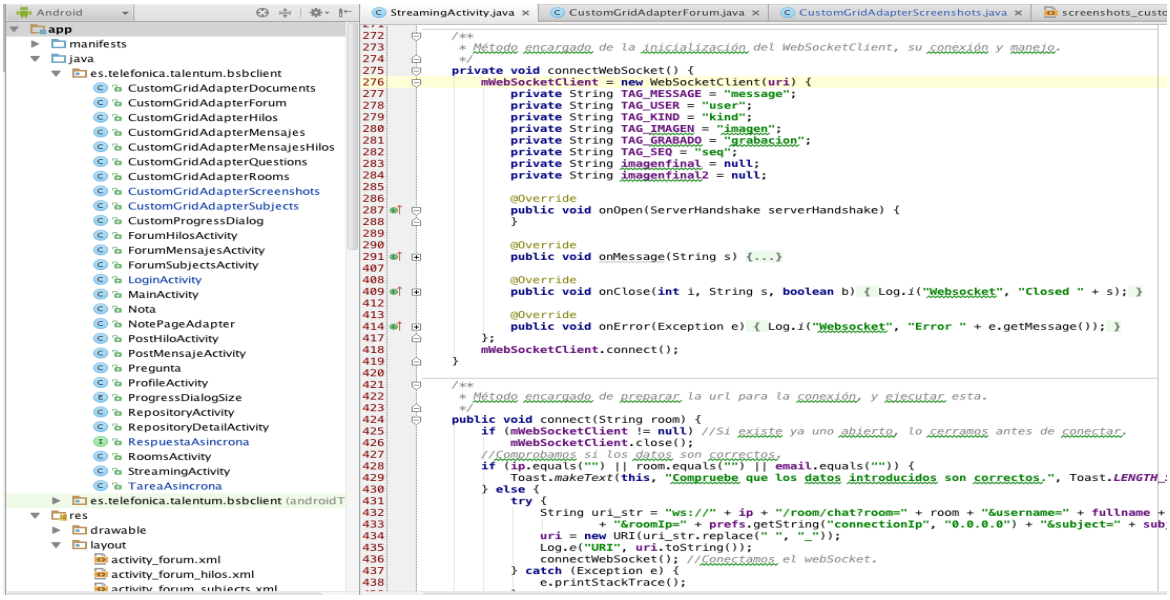


Fig. 3.23 Método *connect* y *connectWebSocket* para la comunicación vía WebSockets en la aplicación Android

Conexión Bluetooth

La conexión Bluetooth utilizada en este proyecto es necesaria para la transmisión de audio del profesor a la aplicación del profesor. Para ello se desvía el micrófono y audio del dispositivo

Android al que posee el dispositivo Bluetooth. Android proporciona la API de Bluetooth que nos permite realizar ese desvío, y además gestionar la conexión de dispositivos. La clase *BTReceiver* se encarga de ello, mostrada en la Figura 3.24 y su respuesta visual en la Figura 3.25.

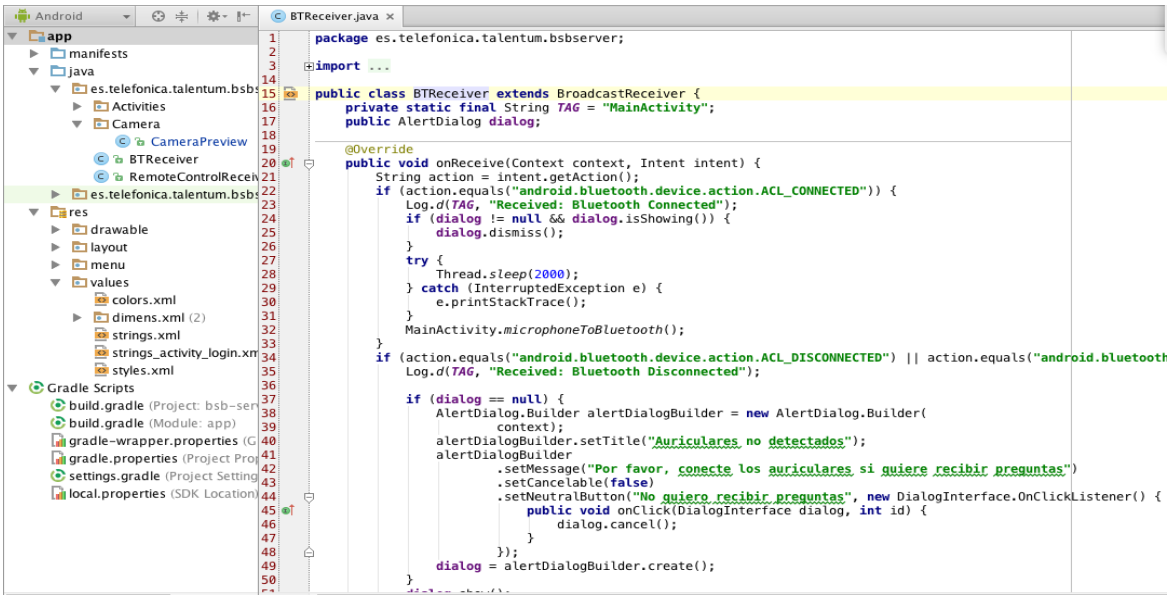


Fig. 3.24 Clase *BTReceiver* para la comunicación vía Bluetooth en la aplicación Android

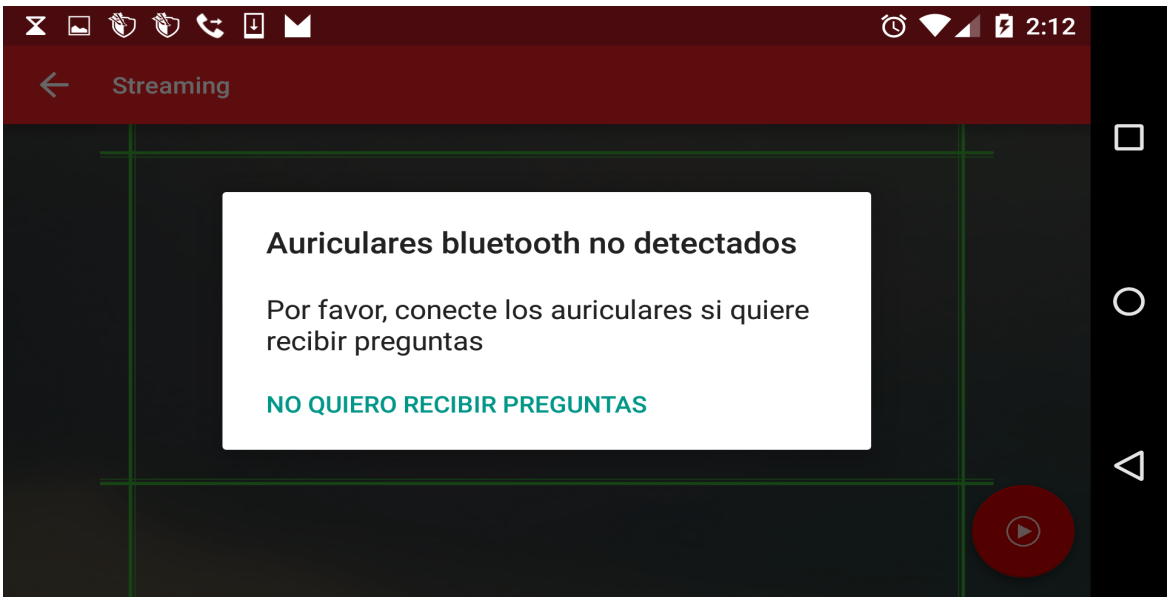


Fig. 3.25 Respuesta visual en la aplicación

Conexión con la base de datos

Para la comunicación del servidor y la base de datos se utiliza la API de Java JDBC. Con ella se consigue establecer las peticiones a la base de datos para recuperar la información necesaria para controlar el modelo del servidor. Esto se consigue indicando donde reside la base de datos, su usuario y contraseña en el archivo de configuración *Application*, como se muestra en la Figura 3.26.

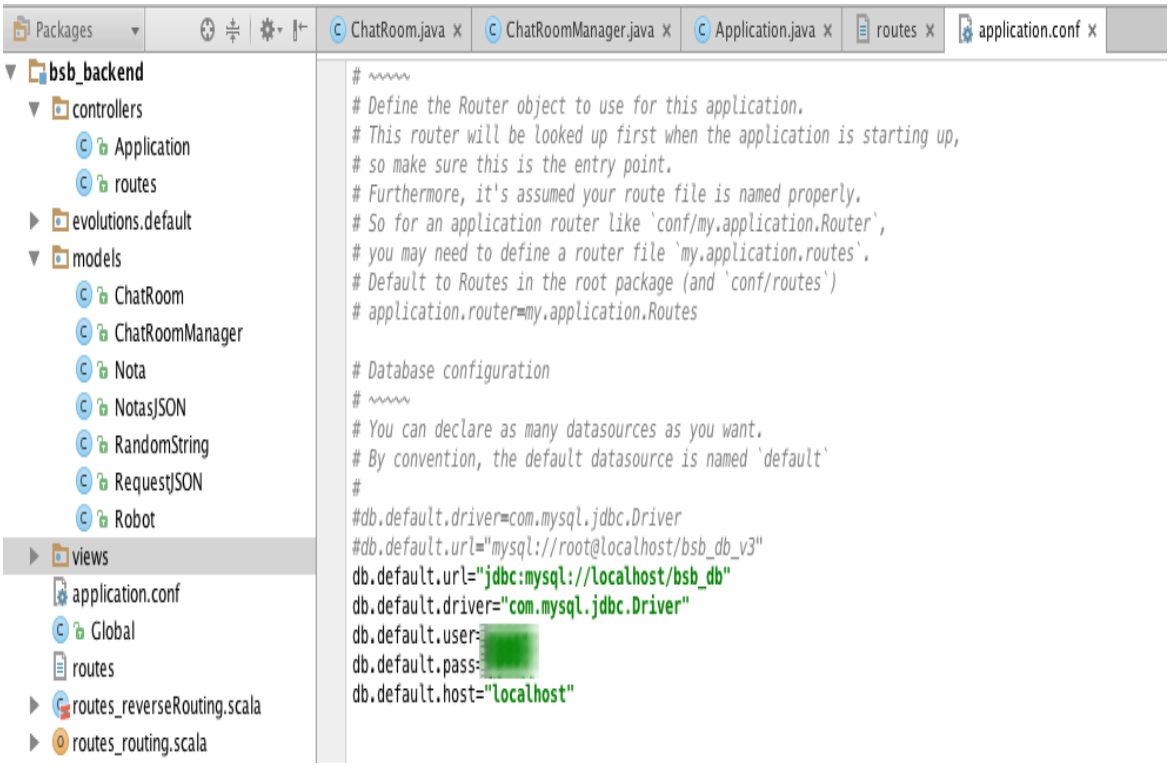


Fig. 3.26 Conexión del servidor con la base de datos

3.3.3 Módulo login

La implementación de este módulo se compone de la creación de las vistas de las aplicaciones de Android y de la encriptación llevada a cabo en el servidor para dar seguridad a las contraseñas.

Vista de login

Ambas aplicaciones de Android comparten la misma vista del login, ya que es la primera vista cargada tras iniciar la aplicación. Para la implementación de esta, se utiliza el archivo XML *activity_login*, donde se desarrolla una anidación de varios *RelativeLayout* y *LinearLayout* para conseguir la posición de los elementos del cuadro de login, como se muestra en la Figura 3.27.

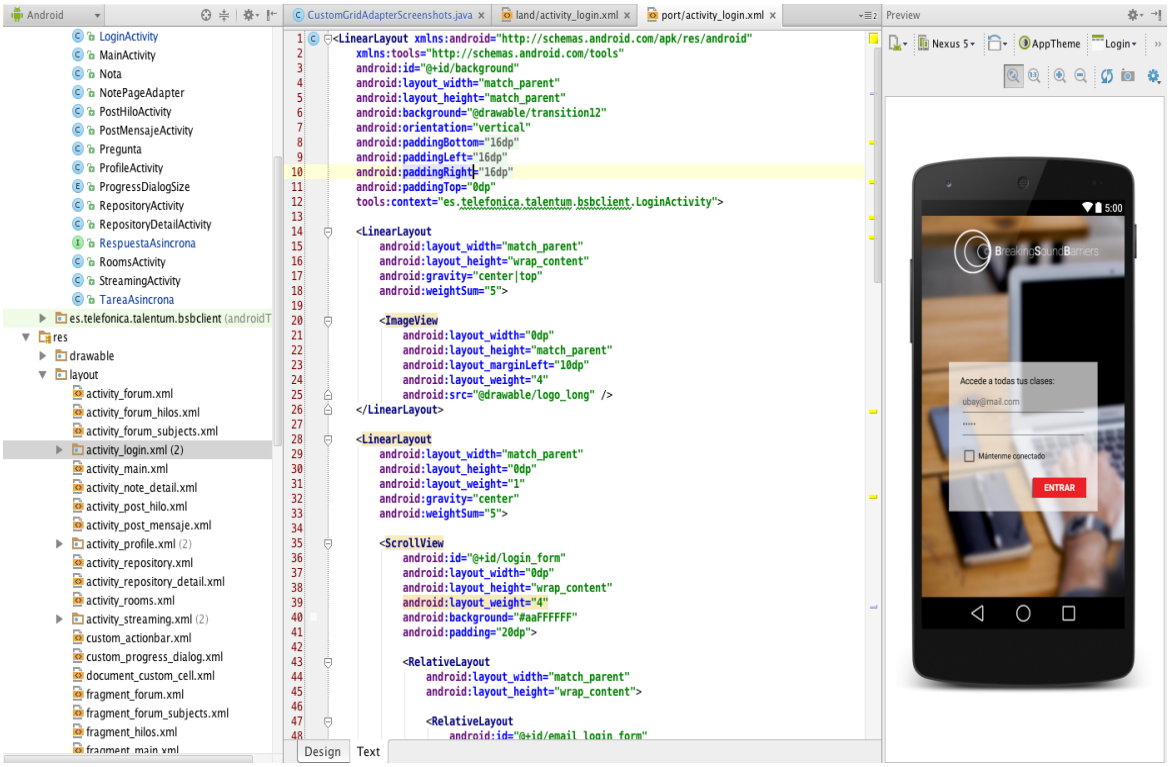


Fig. 3.27 Creación de la interfaz de usuario mediante XML

Cabe destacar que para la implementación de la función de recordar contraseña, se utiliza la API de Android de almacenamiento persistente mediante el uso de la clase textitSharedPreferences. Con ella, si el usuario se ha identificado correctamente en el sistema y tiene marcado la opción de recordar contraseña, sus datos quedarán almacenados en el almacenamiento privado de la aplicación. Esto nos permite que al siguiente inicio de la aplicación, nada más entrar en la vista de login, comprobemos si existen esos datos y proceder automáticamente a realizar la identificación con el servidor, como se muestra en la Figura 3.28.

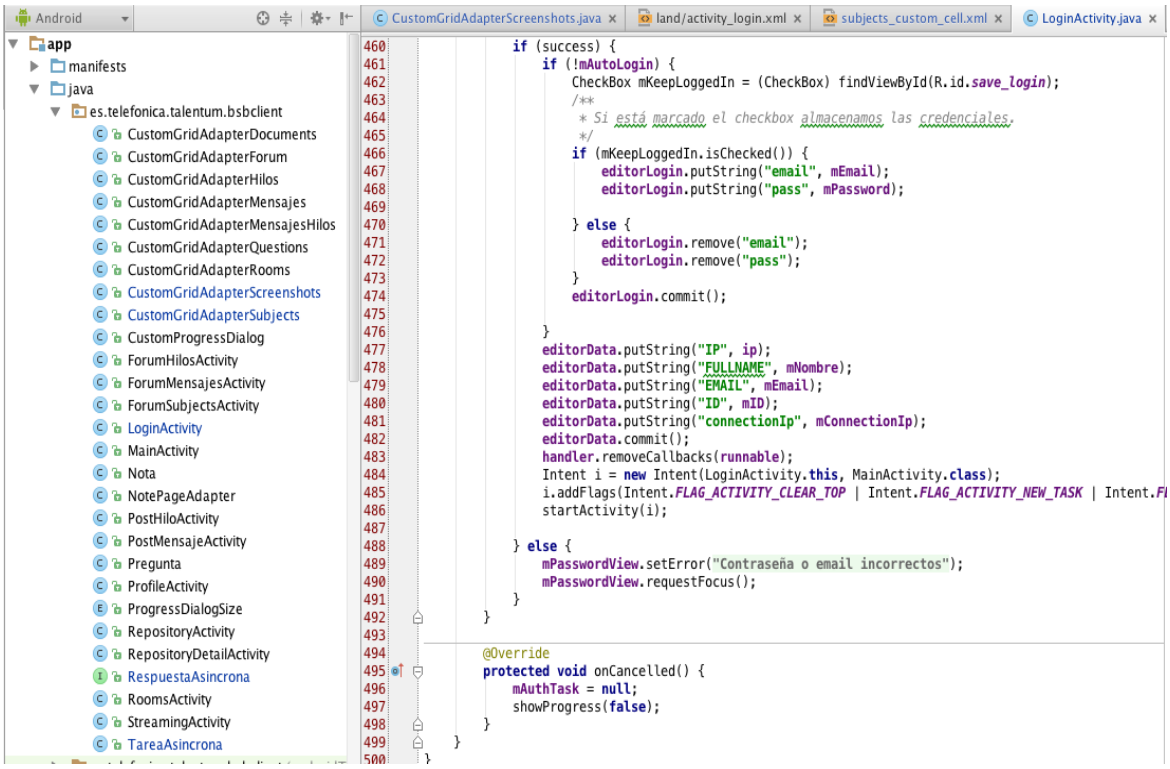


Fig. 3.28 Recordar contraseña mediante el almacenamiento persistente

Encriptación de datos

La encriptación de las contraseñas es llevada a cabo en el servidor. Para ello se utiliza la librería Bcrypt, la cual implementa un esquema de Blowfish Hash al estilo OpenBSD. El uso de esta librería es sencillo, primero se genera una contraseña con hash a partir de la original, la cual es almacenada en nuestra base de datos. Tras esto, cuando recibimos la contraseña del usuario para su identificación, le realizamos la misma conversión a contraseña con hash y la comparamos con la almacenada en el servidor, como se muestra en la Figura 3.29.

La identificación de usuarios se realiza en diferentes tablas de la base de datos. Esto depende si la identificación es de un alumno o un profesor, ya que son dos tipos de usuarios con funcionalidades diferentes dentro del sistema.

```
     * Método de login
     */
    public static Result login() {
        Logger.debug("Intento de conexión desde la ip "+request().remoteAddress().toString());
        JsonNode json = request().body().asJson();
        String id = json.findPath("id").textValue();
        Connection conn = null;
        Statement stmt = null;

        ObjectNode jsonResult = Json.newObject();
        jsonResult.put("connection_ip",request().remoteAddress().toString());
        if (json.size() == 0) {
            return badRequest("0");
        } else {
            try {
                String email = json.findPath("email").textValue();
                String pass = json.findPath("pass").textValue();
                String professor = json.findPath("professor").textValue();

                conn = DB.getConnection();
                stmt = conn.createStatement();

                String sql;
                String buscar_id;
                if (professor.equals("0")) {
                    sql = "SELECT * FROM ALUMNO WHERE email='" + email + "'";
                    buscar_id="ID_Alumno";
                } else {
                    Logger.debug("Intento de conexión a la tabla profesor");
                    sql = "SELECT * FROM PROFESOR WHERE email='" + email + "'";
                    buscar_id="ID_Profesor";
                }

                ResultSet results = stmt.executeQuery(sql);
                conn.close();
                while (results.next()) {
                    if (BCrypt.checkpw(pass, results.getString("password"))) {
                        jsonResult.put("fullname", results.getString("fullname"));
                        jsonResult.put("ID", results.getString(buscar_id));
                        jsonResult.put("success", "yes");
                    } else {
                        jsonResult.put("success", "no");
                    }
                }
            } catch (Exception e) {
                Logger.error("Error: " + e);
                jsonResult.put("success", "no");
            }
        }
        return ok(jsonResult);
```

Fig. 3.29 Método que comprueba la identificación en el servidor

3.3.4 Módulo vista principal

La implementación de este módulo es similar en las dos aplicaciones Android. En ambas se implementa un listado de tarjetas, mientras que en la aplicación del profesor se añade la opción de crear nuevas asignaturas.

Listado de tarjetas

En la aplicación del alumno se muestran cinco tarjetas fijas de información que enseñan la funcionalidad del resto de vistas de la aplicación, más diferentes tarjetas variables que recuperan información de la actividad reciente generada por el usuario dentro del sistema.

Por otro lado, en la aplicación del profesor, las tarjetas muestran las asignaturas que posee el profesor, que brindan acceso directo a crear una sesión de streaming.

Ambas acciones de listado siguen el mismo procedimiento. Se realiza una petición al servidor y se recibe un mensaje en formato JSON sobre la lista de elementos a reproducir en la vista. Estos elementos son pasados a un adaptador, que es una clase especial de Android para rellenar los elementos de ListView o GridView contenidos en la vista. Con esto conseguimos rellenar una vista dinámicamente de elementos con un formato fijo, creando la lista de tarjetas, pero con diferente contenido. Como ejemplos de esto están la clase

CustomGridAdapter y el documento XML *subjects_custom_cell*, mostrados en la Figura 3.30 y en la Figura 3.31.

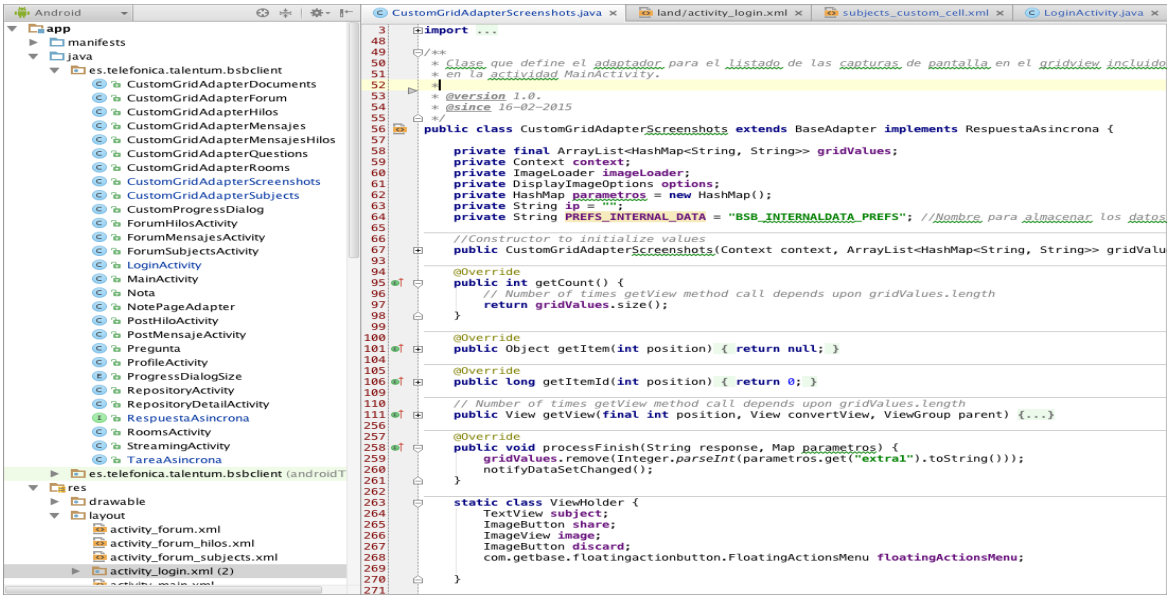


Fig. 3.30 Adaptador de elementos *CustomGridAdapter* de una tarjeta de asignatura de la aplicación del profesor

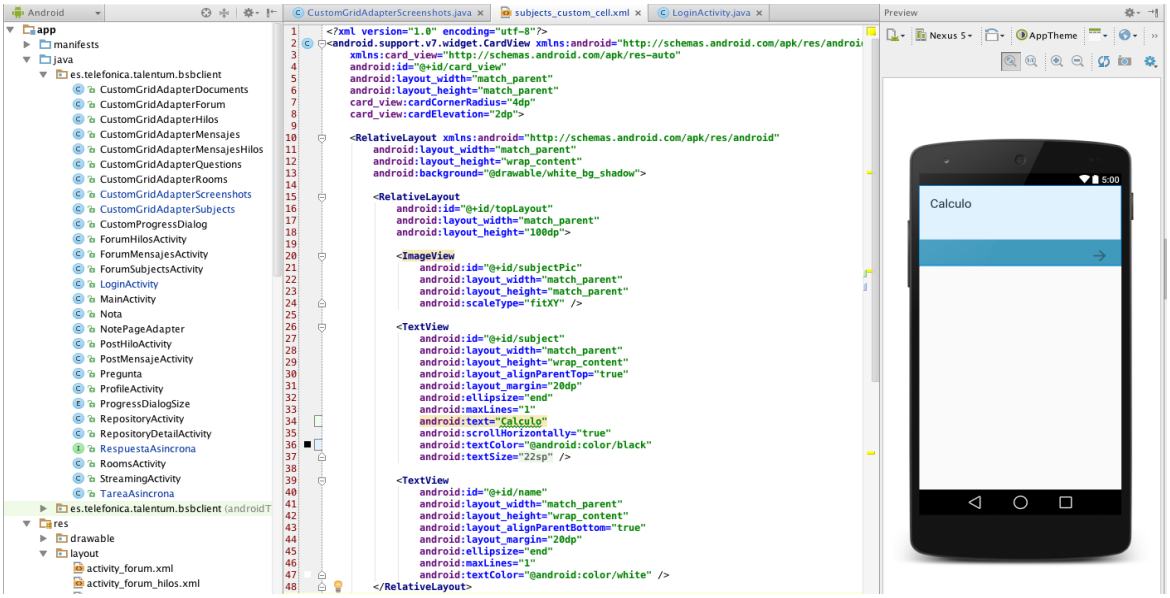


Fig. 3.31 Formato base de una tarjeta de asignatura de la aplicación del profesor con *subjects_custom_cell*

Creación de asignaturas

Para la creación de asignaturas en la aplicación del profesor se lanza una nueva actividad, definida en *SubjectsActivity* y con interfaz diseñada en el documento XML *fragment_subjectsnew*, mostradas en la Figura 3.32 y en la Figura 3.33. Aquí el profesor creará y definirá el nombre y la foto de fondo de la signatura, quedando registrado en el sistema. Cabe destacar que para la toma de la foto de fondo para la asignatura, se utilizan componentes de otras aplicaciones para conseguir la imagen. Según lo que elija el profesor, de entre las opciones de tomar foto o seleccionar desde la galería, se lanzará el método de Android *startActivityForResult* que abrirá la aplicación indicada para conseguir el recurso de la imagen y devolverla a nuestra actividad.

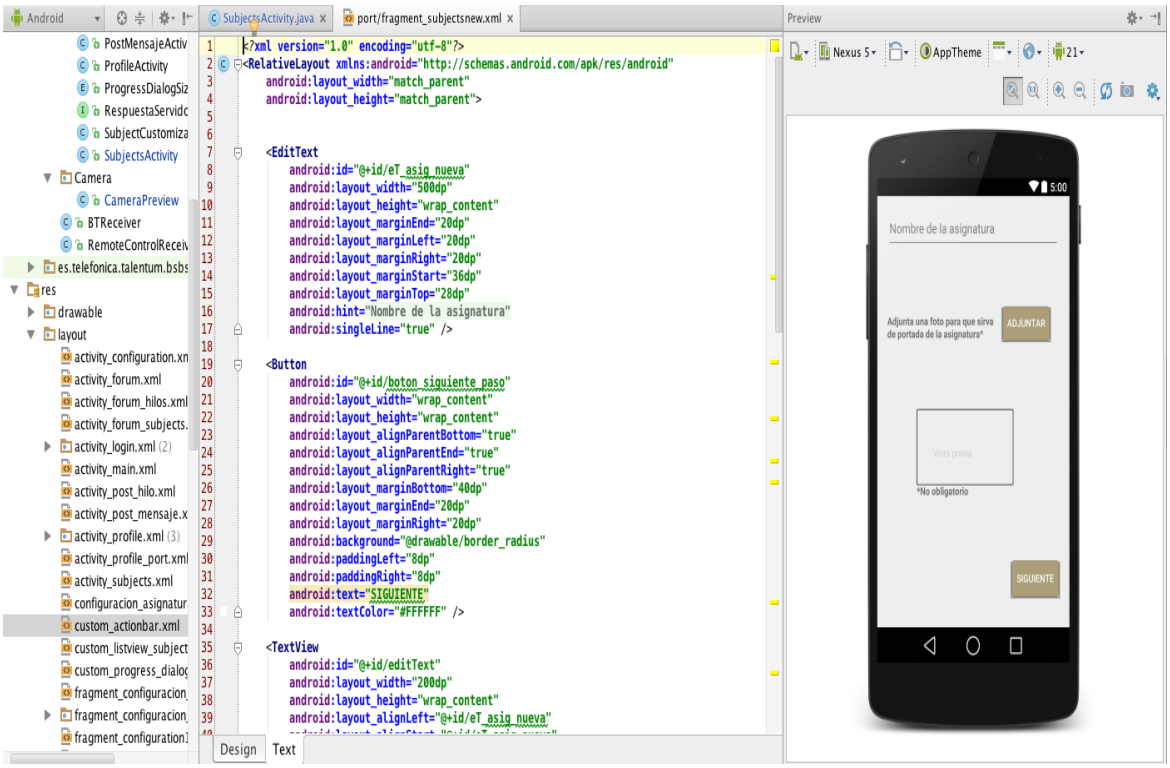


Fig. 3.32 Documento XML *fragment_subjectsnew* que define la interfaz gráfica

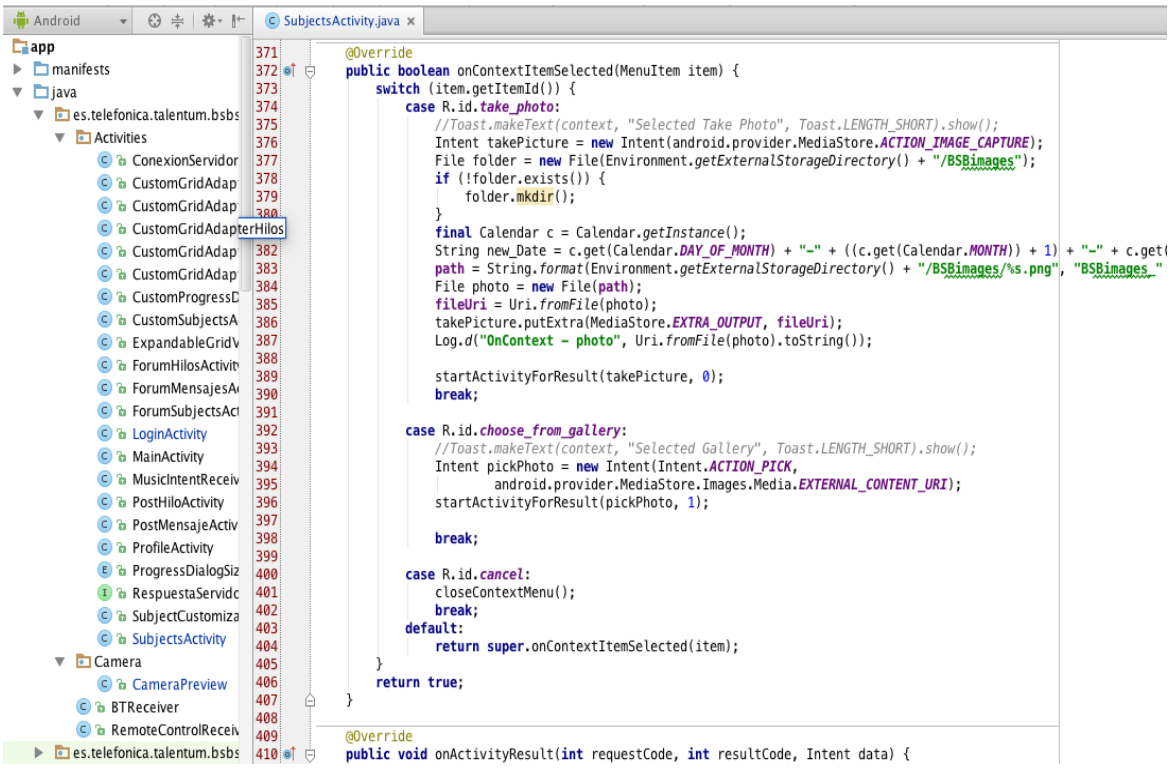


Fig. 3.33 Clase *SubjectsActivity* encargada de la gestión de la creación de asignaturas en la aplicación del profesor

3.3.5 Módulo streaming

La implementación del módulo de streaming dista bastante en ambas aplicaciones Android. Por ello veremos su desarrollo por partes separadas.

Aplicación profesor

El módulo de streaming de la aplicación del profesor está compuesto por dos vistas. La primera definida por el documento XML *fragment_filters*, mostrado en la Figura 3.34, que se encarga de establecer la configuración de la sesión de la asignatura en cuestión, eligiendo entre sesión privada y/o acceso con contraseña. La sesión privada restringirá la conexión de alumnos que no estén dentro de la misma WiFi en la que se encuentre el profesor emitiendo la sesión. Esto se consigue recuperando la dirección pública del dispositivo del profesor, que posteriormente se comparará con la de los dispositivos de los alumnos. Para esto se

utiliza una pagina web externa que nos facilita este dato. Con esto conseguimos que si el profesor lo desea, no se pueda acceder a la sesión si no se encuentra en el lugar de la emisión. La protección con contraseña evita a personas que podrían estar conectados al mismo WiFi acceder a la sesión, no pertenecientes a esa asignatura.

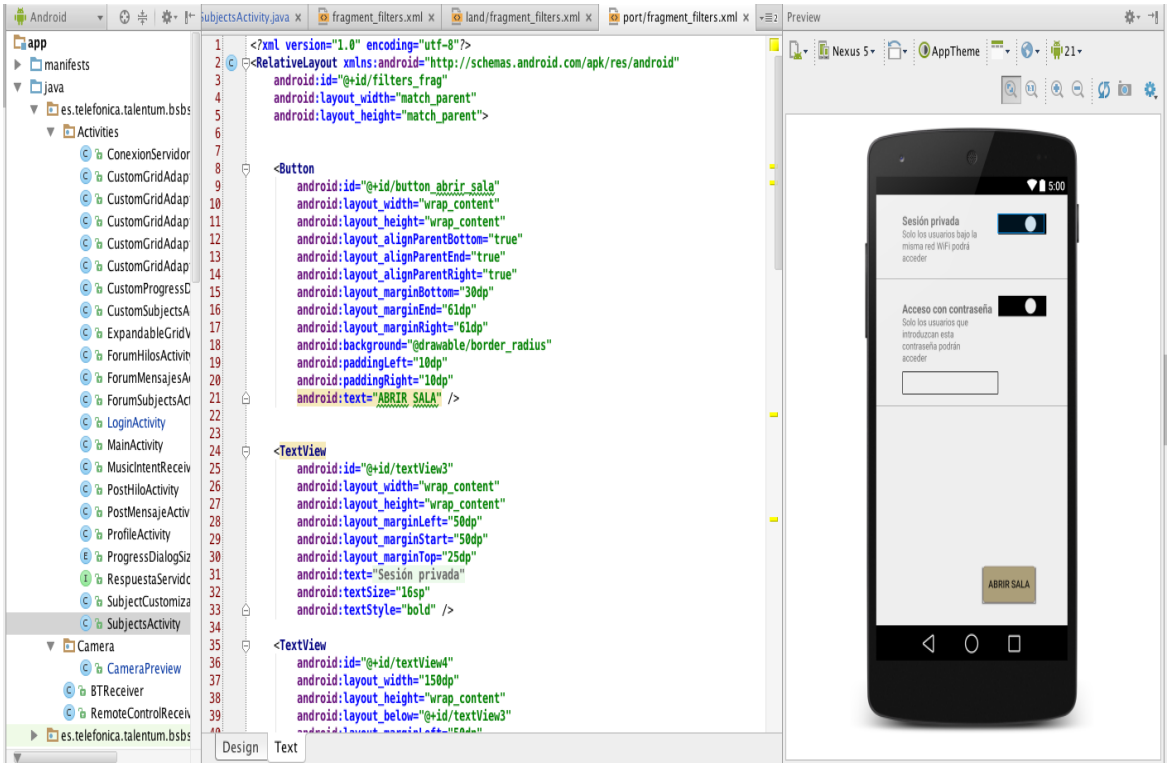


Fig. 3.34 Documento XML *fragment_filters* que define la interfaz gráfica

Tras haber configurado la sesión de streaming a las necesidades del profesor, entramos en la actividad principal de streaming, definida por *activity_main*, mostrada en la Figura 3.35. Al entrar en ella se establece la sala de chat mediante WebSockets en el servidor, y tras comprobar el estado de los dispositivos Bluetooth y pulsando el botón de play se empezará a emitir vídeo y audio por el canal WebSocket:

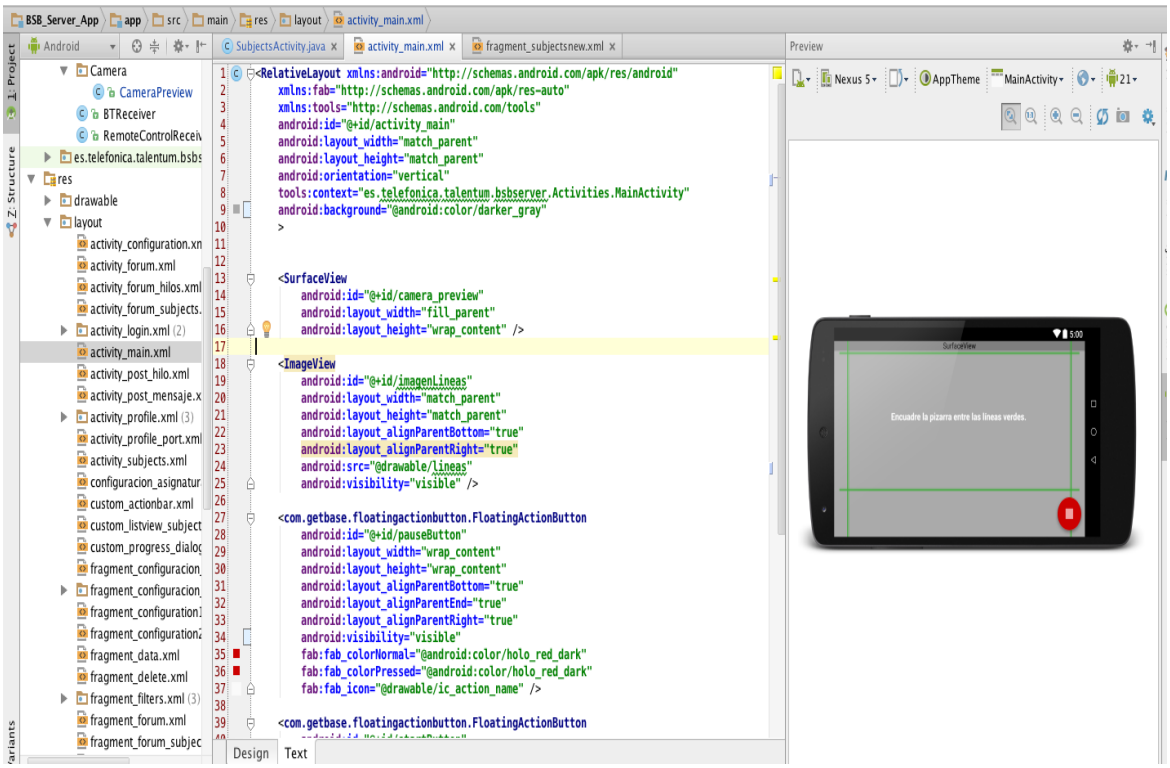


Fig. 3.35 Documento XML *activity_main* que define la interfaz gráfica

Al entrar en la vista de streaming configuramos la vista previa de la cámara. Esto se consigue gracias a la API de la cámara de Android implementada en la clase *CameraPreview*, mostrada en la Figura 3.36.

Para la implementación de la grabación de vídeo no existe ninguna API oficial de Android que permita grabar vídeo en tiempo real y codificarlo para poder enviarlo por la red. Tras la prueba con diferentes librerías de GitHub de código abierto, en la que los resultados de calidad no eran convincentes y pixelaban con gran frecuencia, haciendo imposible la visualización. Se opta por la creación de emisión de vídeo utilizando la tecnología Motion JPG.

Motion JPG [35] es una codificación para el vídeo en el que cada fotograma es una imagen comprimida en formato JPEG. Tomadas las imágenes en una secuencia continuada representan el vídeo.

Gracias al método *previewCallback* proporcionado por la API de la cámara de Android, conseguimos que cada vez que se produzca un fotograma de visualización previa de la

cámara, obtengamos esa imagen para poder procesarla, comprimirla a formato JPEG y poder mandarla al servidor.

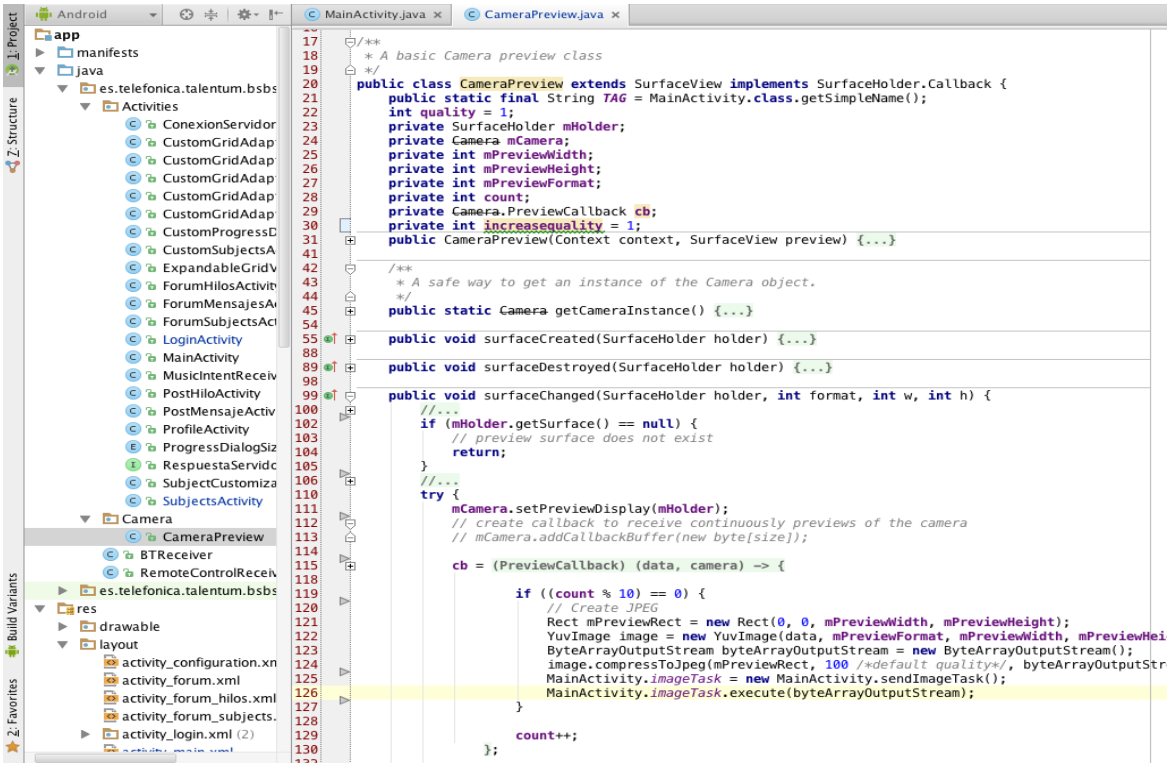


Fig. 3.36 Clase *CameraPreview* encargada de gestionar la cámara en la aplicación Android

Paralelamente al grabado del vídeo se realiza el reconocimiento de voz del profesor. Gracias a la API de Android de speech-to-text, podemos implementar la trascripción de audio a texto. Con la clase *SpeechRecognizer* e implementado los métodos de la clase *RecognitionListener*, mostrados en la Figura 3.37, podemos obtener los resultados del motor de reconocimiento de audio de Android. Además esta clase nos permite reconocer parcialmente los resultados interpretados por el motor de reconocimiento, así evitando esperar a un resultado global tras la finalización del discurso, proporcionando un aspecto más parecido a una traducción en tiempo real.

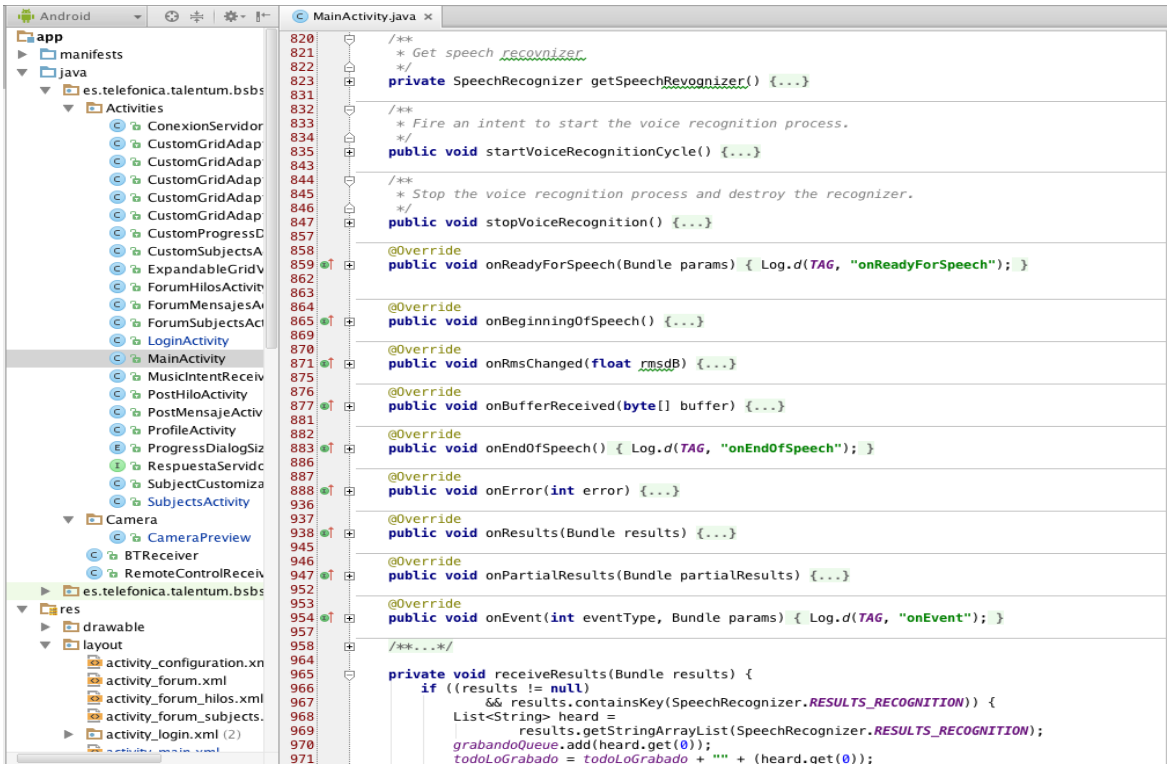


Fig. 3.37 Métodos implementados de la clase *RecognitionListener* encargada del reconocimiento de audio

Por último, para que el profesor pueda recibir las preguntas formuladas, se hace uso también del motor de reconocimiento de audio, mostrado en la Figura 3.38 los métodos utilizados . Las preguntas llegan a la aplicación del profesor mediante una notificación sonora, por el canal WebSocket establecido y se almacenan en una pila. Tras esto, el profesor podrá activar el botón de acción del dispositivo Bluetooth y se le reproducirá el número de preguntas que le esperan por reproducir. El profesor tendrá las opciones de decir las palabras *siguiente*, *anterior* o *repetir*, o con los botones del dispositivo Bluetooth, para manejarse en la reproducción de las preguntas.

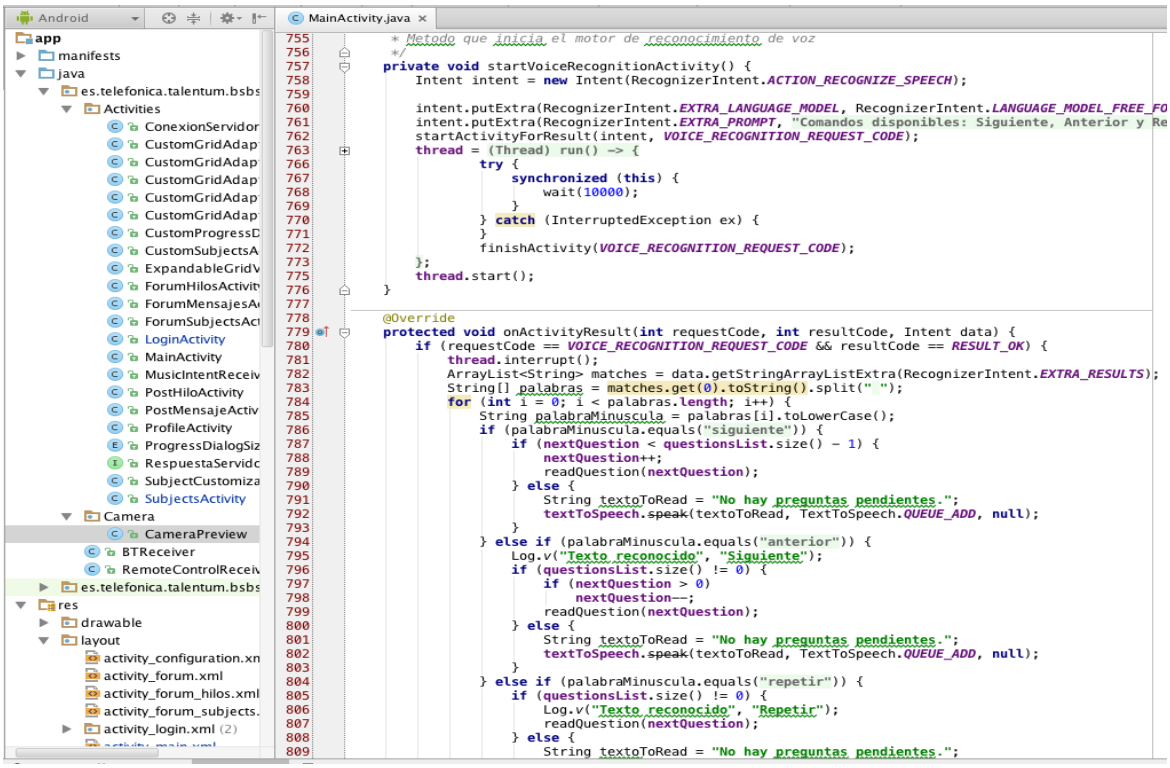


Fig. 3.38 Métodos encargados del reconocimiento de audio para el manejo de reproducción de preguntas

Aplicación alumno

La implementación del módulo de streaming de la aplicación del alumno consta de dos vistas. La primera definida por el documento XML *fragment_rooms*, mostrado en la Figura 3.39, lista las sesiones abiertas a las que el alumno puede unirse. Tras la selección de la sesión y posible introducción de contraseña, se establece la conexión vía WebSocket a la sesión creada por el profesor.

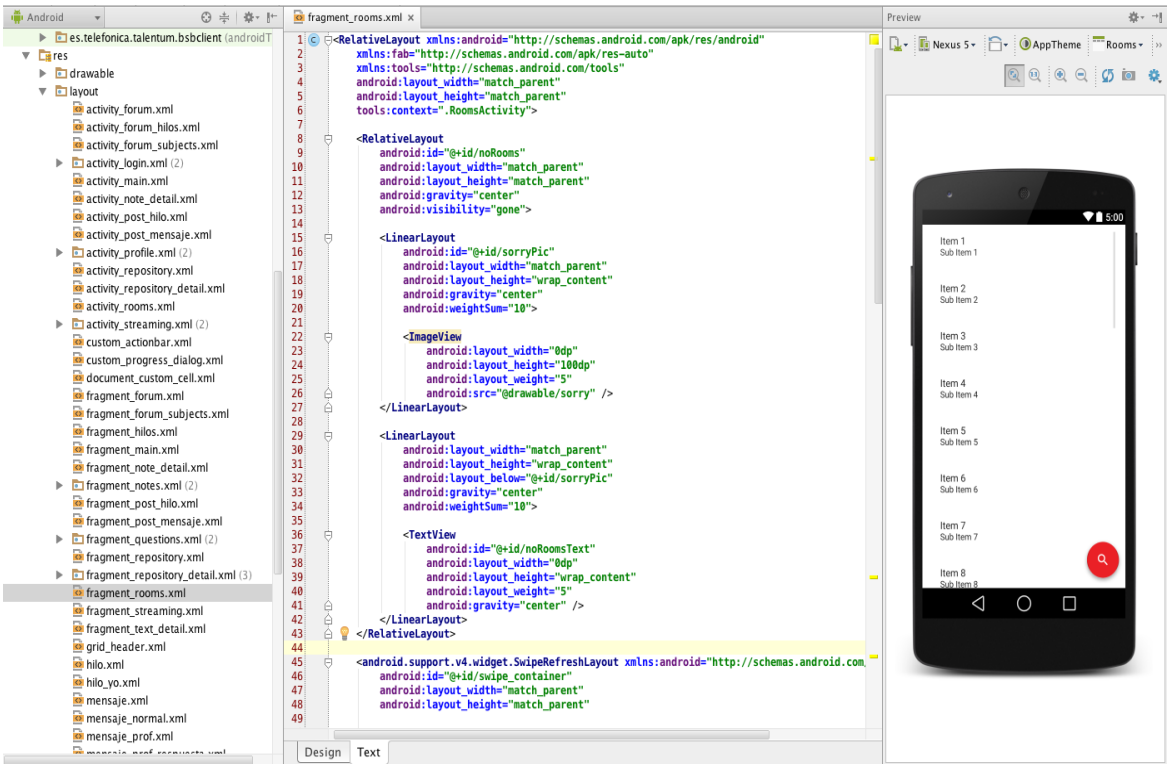


Fig. 3.39 Documento XML *fragment_rooms* que define la interfaz gráfica

Una vez dentro de la sesión, se muestra la vista principal de streaming, mostrada en la Figura 3.40. Está compuesta por tres fragmentos diferentes definido en los documentos XML *fragment_streaming*, *fragment_notes* y *fragment_questions*. Estos dos últimos se alternan visiblemente en la parte derecha de la vista, según se pulse el botón de pregunta. *fragment_streaming* se encarga de mostrar el vídeo y los subtítulos. Mientras que *fragment_notes* se encarga de recoger los apuntes del alumno y *fragment_questions* de gestionar las preguntas.

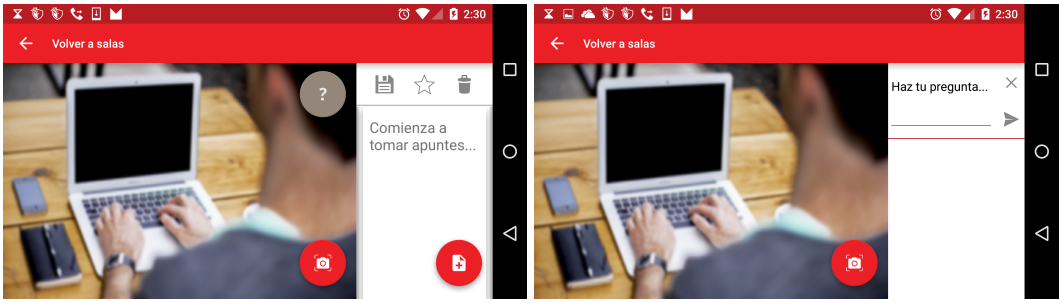


Fig. 3.40 Vista streaming alumno

El vídeo se reproducirá en el fragmento *fragment_streaming*. La aplicación del alumno recibirá por el canal WebSocket un mensaje en formato JSON con la información de un

fotograma, y gracias a la clase *setImageTask*, mostrada en la Figura 3.41, la cual extiende de la clase asíncrona *AsyncTask*, decodificaremos el fotograma convirtiéndolo en una imagen Bitmap. Con el fotograma recuperado, lo podremos mostrar en el fragmento de streaming.

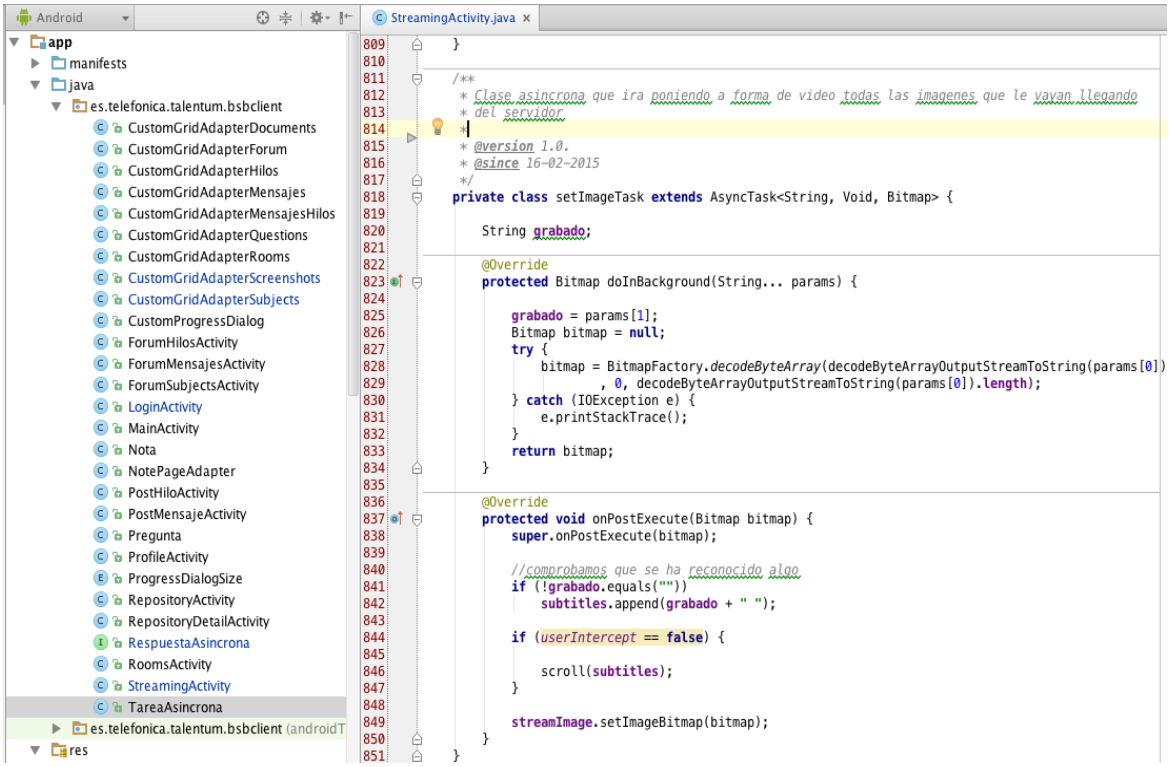


Fig. 3.41 Clase *setImageTask* encargada de gestionar el vídeo y audio durante el streaming

A la par que el vídeo, el audio también se transmite en los mismos mensajes del canal WebSocket con formato JSON, como se muestran en la Figura 3.42 los métodos utilizados. En este caso es más fácil recuperar la trascripción, ya que la información es una cadena de texto, fácilmente recuperable para su representación en el fragmento de streaming. Para dar realismo a la trascripción, y que tengan un formato de subtítulos, con la llegada de nuevo texto el cuadro reservado para la trascripción irá desplazándose automáticamente mostrando el nuevo texto, salvo que el usuario lo esté haciendo manualmente:

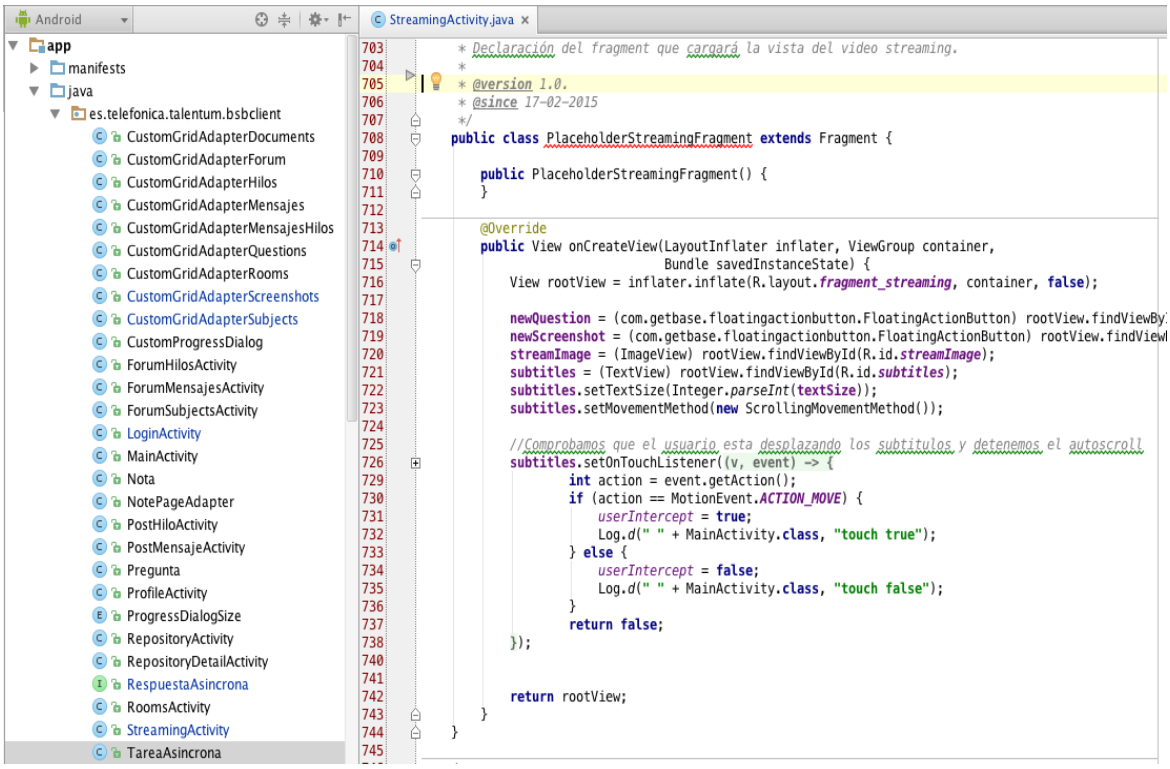


Fig. 3.42 Método encargado de comprobar el movimiento de los subtítulos

La implementación de la toma de notas se hace a través de un adaptador incrustado dentro de una lista de cuadros de edición de texto, como muestran los métodos de la Figura 3.43. Esto nos permite cambiar de nota y editarlas dentro de un mismo espacio visual. La toma de notas contiene tres botones principales: nueva nota, crea un nuevo espacio para tomar una nota, favorito, elige esa nota como la primera a mostrar en el listado de notas, y guardar, para respaldar los cambios hechos en las notas, en el servidor y que puedan estar disponibles en el repositorio. Automáticamente, cada cierto tiempo las notas se guardan en el servidor, en caso de que el usuario olvide guardarlas.

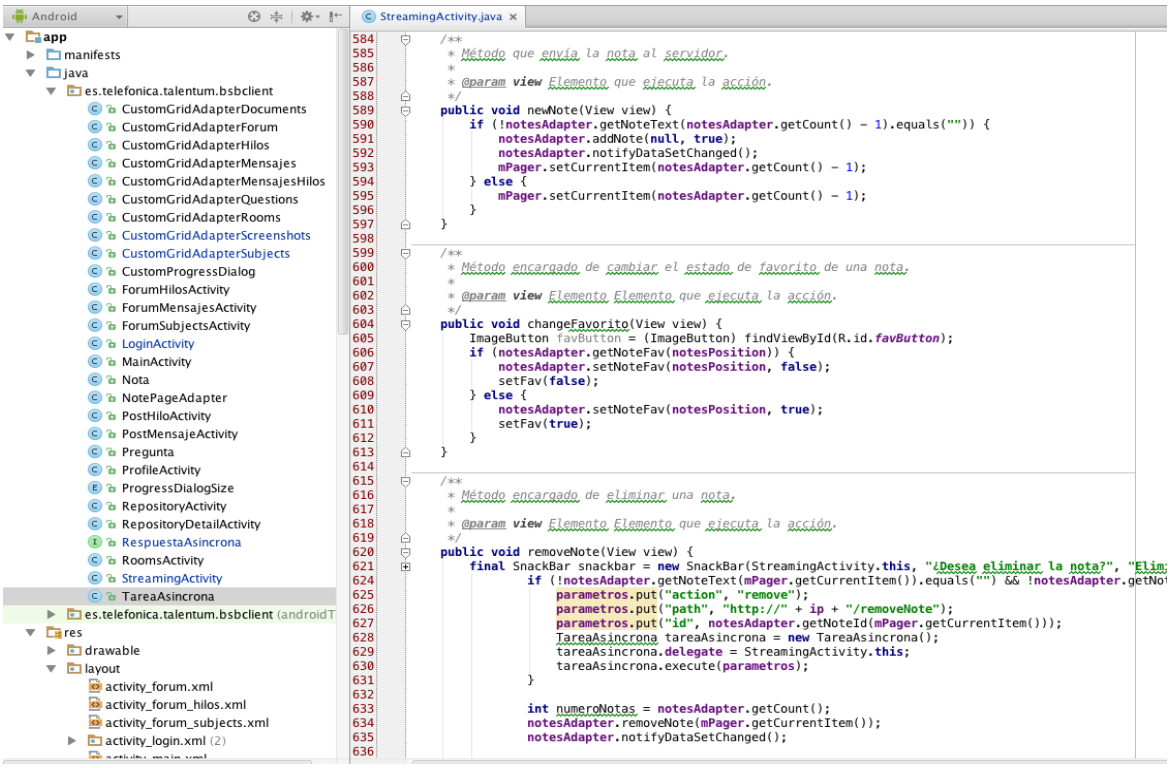


Fig. 3.43 Varios métodos utilizados en la implementación de las notas

El fragmento de preguntas se accede tras pulsar el botón de pregunta y reemplaza a la vista de notas. Desde aquí se muestra un cuadro de texto para formular preguntas y una lista para la votación de otras preguntas realizadas por otros alumnos. Estas preguntas son almacenadas en el servidor hasta que el número de votos sea elevado y se trasmite en forma de notificación al profesor. Con este sistema de votos evitamos saturar al profesor con notificaciones, y hacer llegar las preguntas más relevantes para los alumnos, como muestran los métodos de la Figura 3.44.

```
442
443     /**
444      * Método que muestra el cuadro de texto para el envío de una pregunta.
445      */
446     public void question(View view) {
447
448         newQuestion.setVisibility(View.GONE);
449
450         placeholderQuestionsFragment = new PlaceholderQuestionsFragment();
451         getSupportFragmentManager().beginTransaction()
452                 .add(R.id.container_notes, placeholderQuestionsFragment)
453                 .commit();
454     }
455
456     /**
457      * Método que envía una pregunta.
458      */
459     public void sendQuestion(View view) {
460         EditText input = (EditText) findViewById(R.id.questionsText);
461         if (input.getText().toString().equals("")) {
462             Toast.makeText(StreamingActivity.this, "La pregunta no tiene contenido. No ha sido enviada.", Toast.LE
463         } else {
464             HashMap jsonMap = new HashMap();
465             jsonMap.put("nombre", fullname);
466             jsonMap.put("pregunta", input.getText().toString());
467             JSONObject json = new JSONObject(jsonMap);
468             sendMessage(json.toString(), email, true, false, false);
469             input.setText("");
470         }
471         input.clearFocus();
472         InputMethodManager imm = (InputMethodManager) getSystemService(Context.INPUT_METHOD_SERVICE);
473         imm.hideSoftInputFromWindow(view.getWindowToken(), 0);
474     }
475
476     /**
477      * Método que cierra el cuadro de texto para el envío de una pregunta.
478      */
479     public void closeQuestion(View view) {
480         if (placeholderQuestionsFragment != null)
481             getSupportFragmentManager().beginTransaction().remove(placeholderQuestionsFragment).commit();
482         InputMethodManager imm = (InputMethodManager) getSystemService(Context.INPUT_METHOD_SERVICE);
483         imm.hideSoftInputFromWindow(view.getWindowToken(), 0);
484         newQuestion.setVisibility(View.VISIBLE);
485     }
486
487     /**
488      * Translates String to ByteArrayOutputStream
```

Fig. 3.44 Varios métodos utilizados en la implementación de las preguntas

Como extra adicional a la toma de notas, se implementa la función de acceso directo a la toma de capturas de pantalla, como muestra el método *screenshot* de la Figura 3.45. Para ser más eficientes, la captura de pantalla sólo recoge la imagen del vídeo. Esto se implementa guardando el último fotograma recibido hasta el momento del fragmento de streaming. Además se realiza el guardado en el servidor para la posterior utilización en el repositorio.

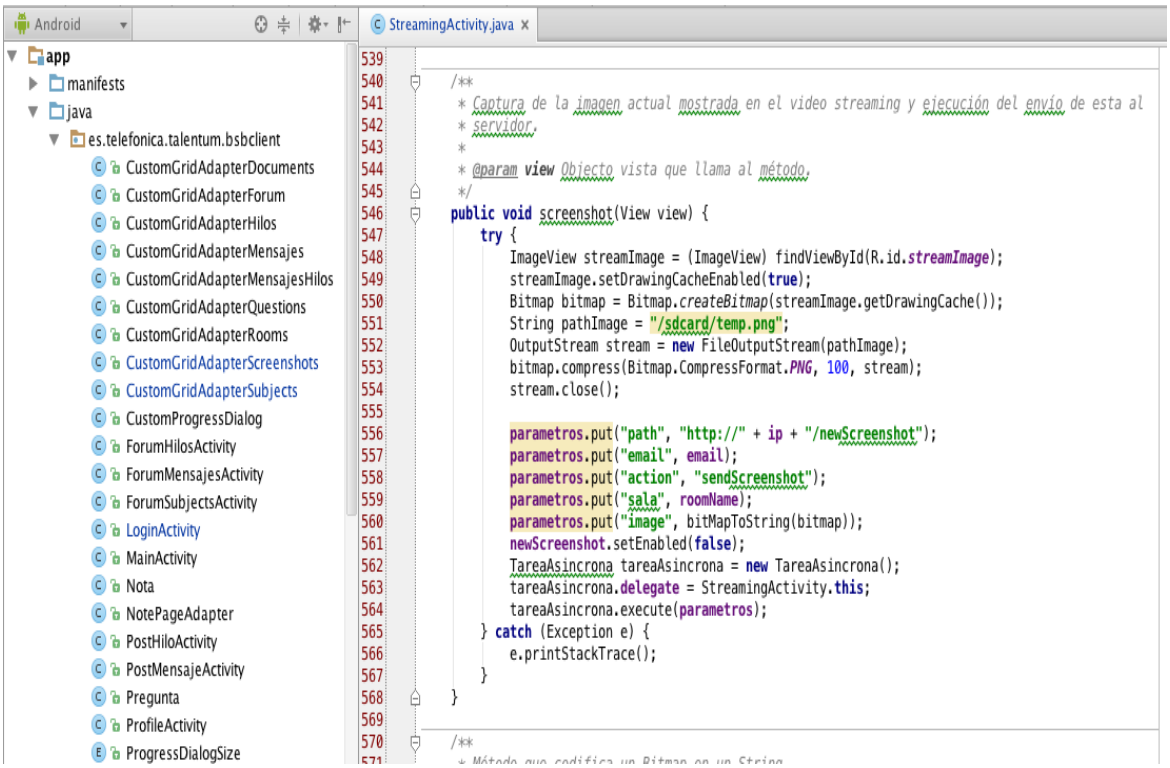


Fig. 3.45 Método *screenshot* encargado de realizar la captura de pantalla y guardarla en el servidor

3.3.6 Módulo foro

El módulo del foro está implementado de igual forma en las aplicaciones del alumno y del profesor. El módulo está desarrollado siguiendo la estructura de vistas siguiente: vista de asignaturas, vista de hilos, vista de mensajes y vista de introducción de mensajes. Todas ellas están relacionadas entre sí a modo de cascada, siendo la primera la vista de asignaturas y la última la de introducción de mensajes.

Vista asignaturas

La implementación de la vista es similar al listado de asignaturas que tiene la vista principal de la aplicación del profesor. La aplicación hará una petición al servidor, preguntando por que asignaturas está suscrito el usuario al foro. El servidor responderá con un listado de asignaturas. En el caso del alumno, se mostrarán las asignaturas en las que este haya

participado en alguna sesión de streaming. En el caso del profesor, se mostrarán todas las asignaturas que imparta. Esta gestión es llevada por la clase *ForumSubjectsActivity*, mostrada en la Figura 3.46.

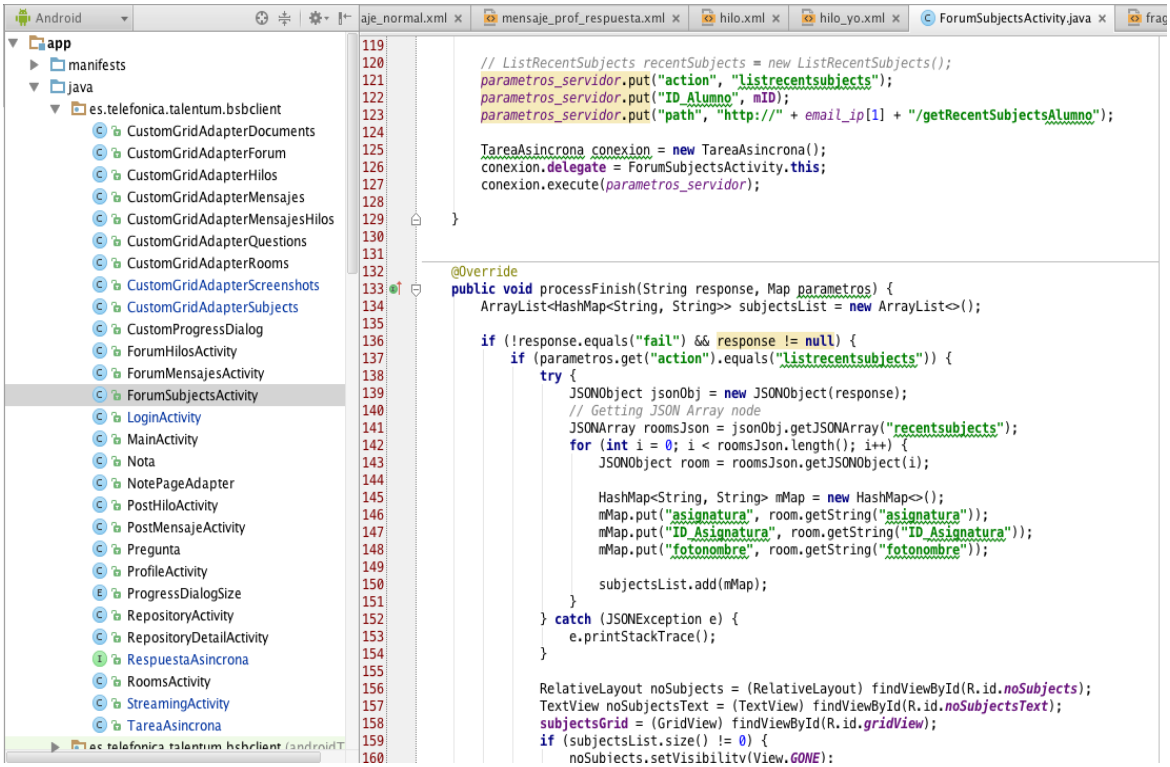


Fig. 3.46 Clase *ForumSubjectsActivity* encargada de gestionar el acceso a las asignaturas del foro

Vista hilo

Una vez dentro de una asignatura, se mostrarán los hilos de discusión que tiene abiertos esta. Entre ellos aparecen diferenciados con diferentes colores, mostrándose por un lado los generados por el sistema, cuando se realiza una pregunta en la sesión de streaming, los creados por los profesores, los creados por los alumnos y los creados por el usuario actual del foro. Con esto se consigue una identificación rápida de los tipos de hilos de discusión que existe en el foro. A ejemplo se encuentra el documento XML *hilo_yo*, mostrado en la Figura 3.47 que define el formato de los hilos generados por el usuario actual.

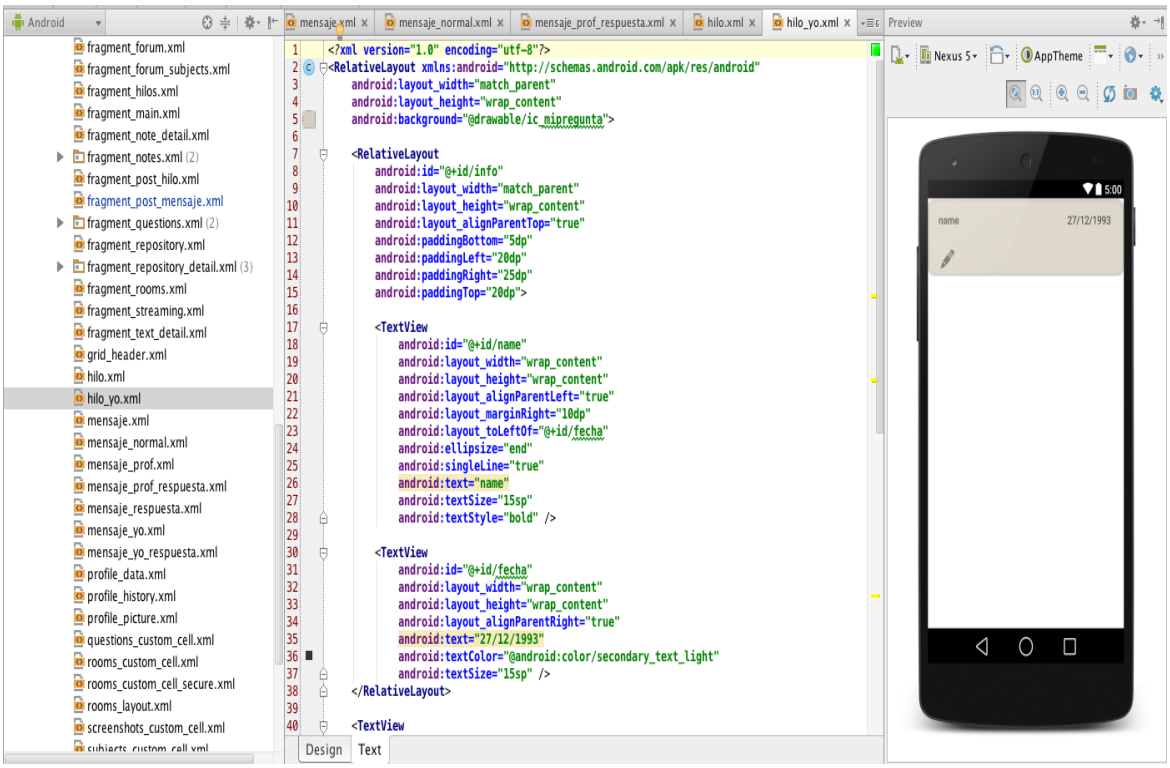


Fig. 3.47 Documento XML *hilo_yo* que define la interfaz gráfica de un hilo de usuario actual

Vista mensajes

Cuando entramos dentro de un hilo, pasamos a mostrar los mensajes que contiene el hilo en otra vista. Esta vista se compondrá por un título fijo, que mostrará el título del hilo donde nos encontremos, y a continuación los mensajes que contiene el hilo. Al igual que los hilos, los mensajes están diferenciados. Se diferencian en color y forma. La forma varía dependiendo si el mensaje lo ha enviado el usuario actual u otro usuario del sistema o dependiendo si es una respuesta a un mensaje anterior o no, consiguiendo así una anidación de mensajes, típica en los foros. Los colores de los mensajes varían igual que los hilos, dependiendo del tipo de usuario que los haya generado. Un mensaje tipo de un usuario diferente al actual se muestra en el documento XML *mensaje*, mostrado en la Figura 3.48.

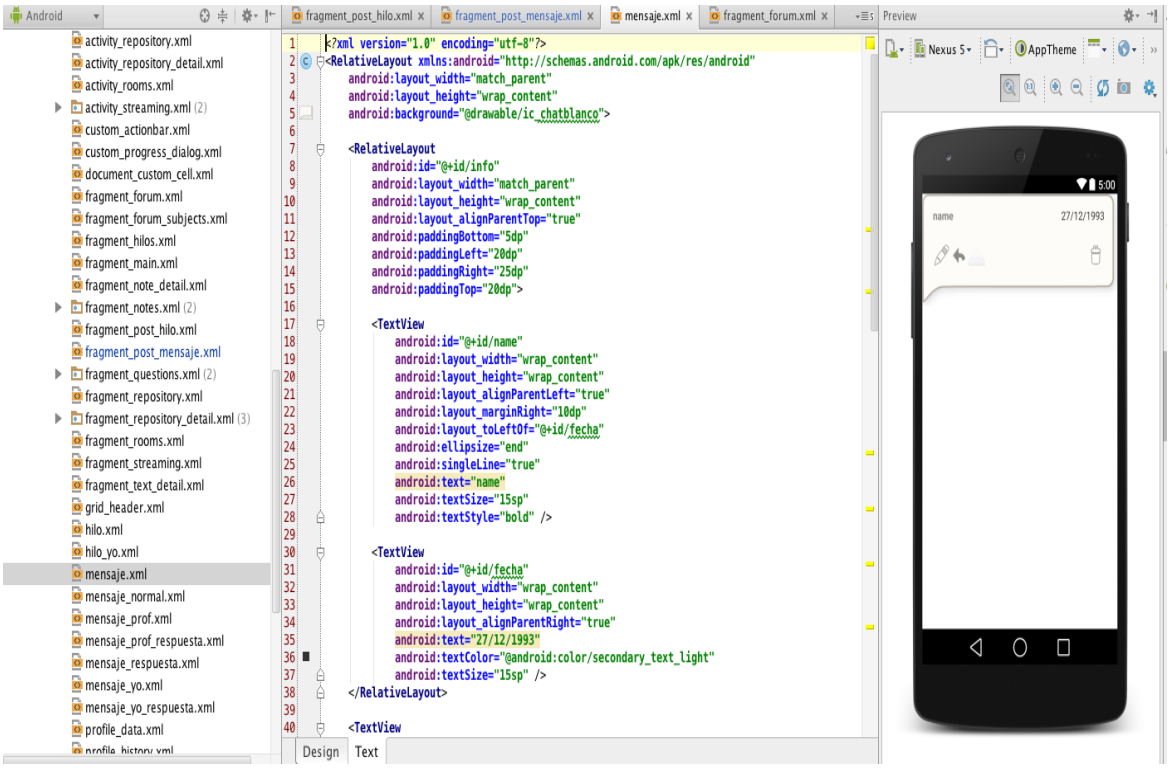


Fig. 3.48 Documento XML *mensaje* que define la interfaz gráfica de un mensaje de usuario diferente al actual

Vista introducción de mensajes

Para introducir, responder o editar mensajes dentro de un hilo se dispone de una vista adicional definida por el documento XML *fragment_post_mensaje*, mostrado en la Figura 3.49. Dependiendo del botón pulsado en la vista anterior, si se ha pulsado respuesta, edición o nuevo mensaje, esta vista actuará creando un mensaje nuevo que se actualizará en la base de datos del servidor.

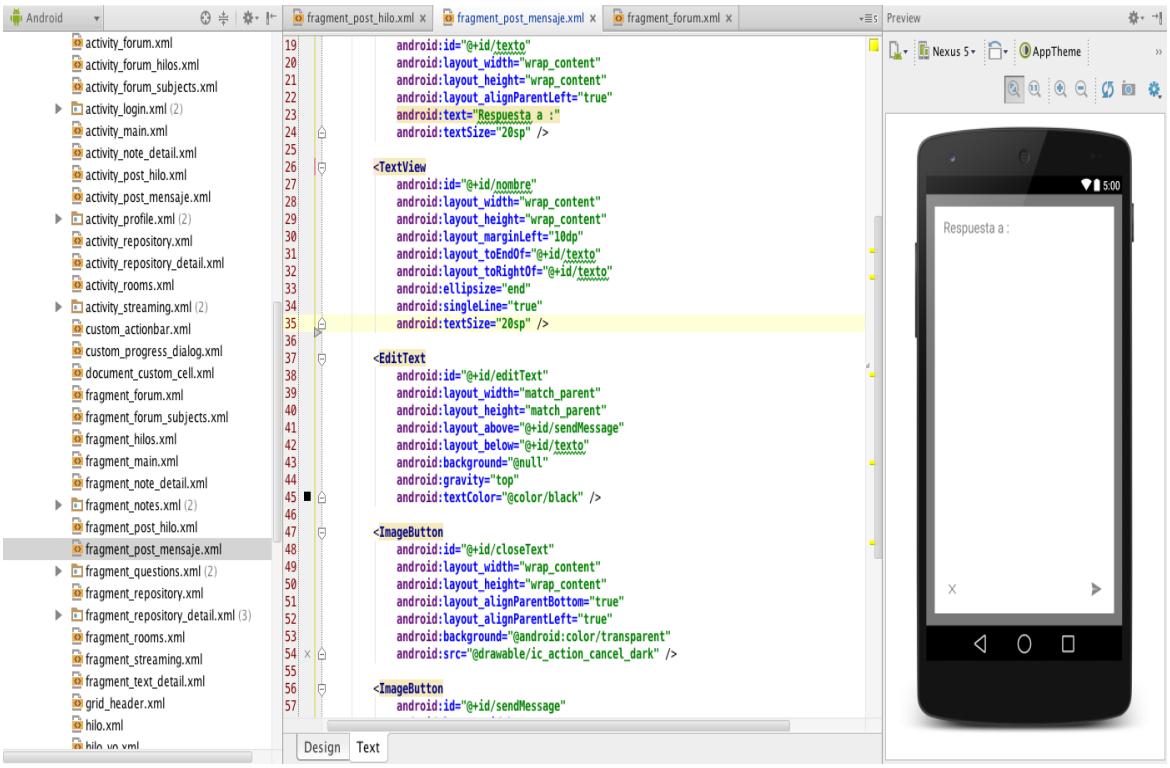


Fig. 3.49 Documento XML *fragment_post_mensaje* que define la interfaz gráfica de la vista introducción de mensajes

3.3.7 Módulo repositorio

El módulo de repositorio sólo está presente en la aplicación del alumno, ya que este es el que necesita recuperar los datos generados en una sesión de streaming. La implementación del módulo del repositorio esta basada en la creación de dos vistas. La primera mostrará una lista de asignaturas, en la que aparecerán sólo las asignaturas en las que el alumno haya participado en una sesión de streaming, al igual que se realiza en el módulo del foro.

La segunda vista comprende el material que se ha generado durante las sesiones de streaming para esa asignatura. Los contenidos incluyen capturas de pantalla, trascripción del profesor y apuntes tomados durante una sesión de streaming. Los contenidos estarán ordenados y diferenciados por fecha de las sesiones de streaming realizadas en esa asignatura. El documento XML *fragment_repository_detail*, mostrado en la Figura 3.50, define las vistas donde será contenida la diferente información del repositorio.

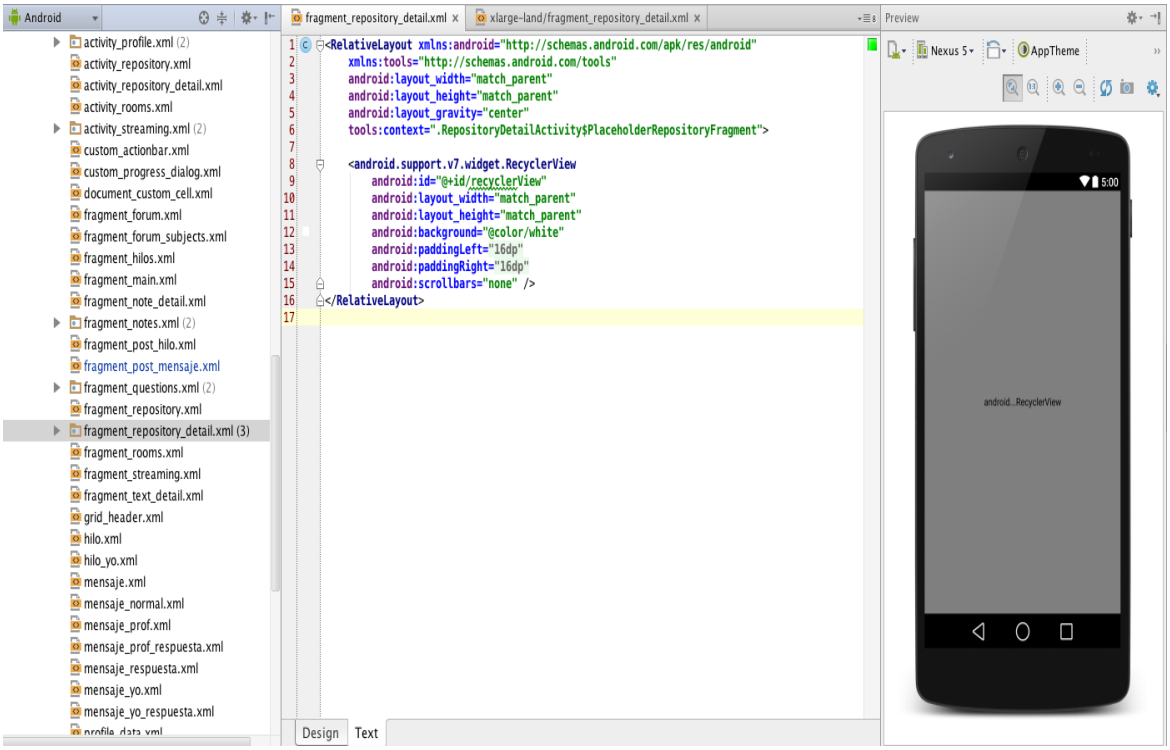


Fig. 3.50 Documento XML *fragment_repository_detail* que define la interfaz gráfica de la vista repositorio

3.3.8 Módulo vista usuario

El último módulo en implementarse es el de la gestión de los datos del usuario. El desarrollo de este módulo es similar en ambas aplicaciones, salvando ciertas partes de configuración propias para cada aplicación. La vista de usuario está compuesta por tres secciones principales. La primera es la foto de usuario, que permite añadir o modificar una fotografía de avatar asociada al usuario. Tras esta, están la modificación de datos de usuario y el borrado de historial. Adicionalmente, en la aplicación del profesor se encuentra una sección para modificar las asignaturas.

Modificación de datos

La sección de modificación de datos permite cambiar el nombre de usuario y la contraseña. Al igual que en el login, el cifrado de la contraseña se realiza en el servidor, antes de almacenarse en la base de datos. El documento XML *profile_data*, mostrado en la Figura 3.51, define la sección de modificación de datos dentro de la vista de usuario.

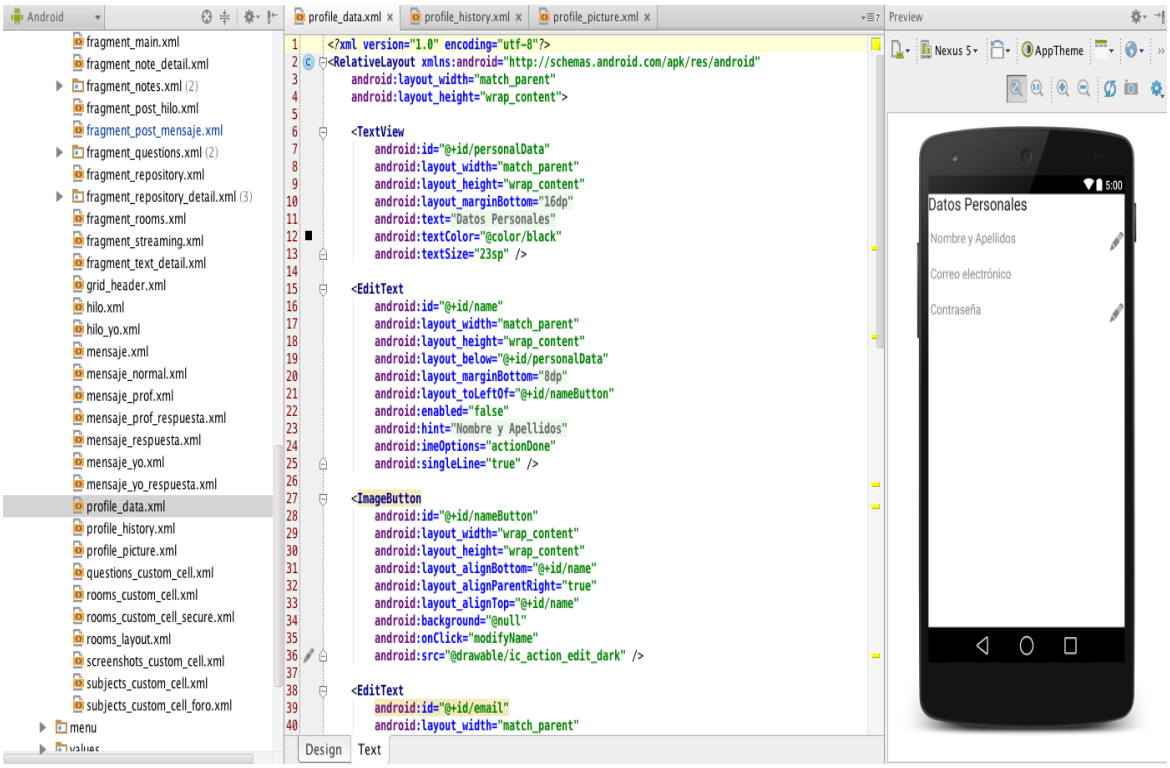


Fig. 3.51 Documento XML *profile_data* que define la interfaz gráfica de modificación de datos

Modificación de historial

La sección de modificación de historial permite eliminar los registros en el sistema del usuario. Entre ellos se encuentra el borrado del contenido, del repositorio o de las asignaturas que se ha participado, con ello también restringiendo el acceso al foro de dichas asignaturas. Todos estos cambios se respaldan en el servidor, modificando la visibilidad de las asignaturas o documentos en la base de datos, para una posible recuperación de la información en caso accidental. El documento XML *fragment_delete*, mostrado en la Figura 3.52, define la sección de modificación de historial dentro de la vista de usuario.

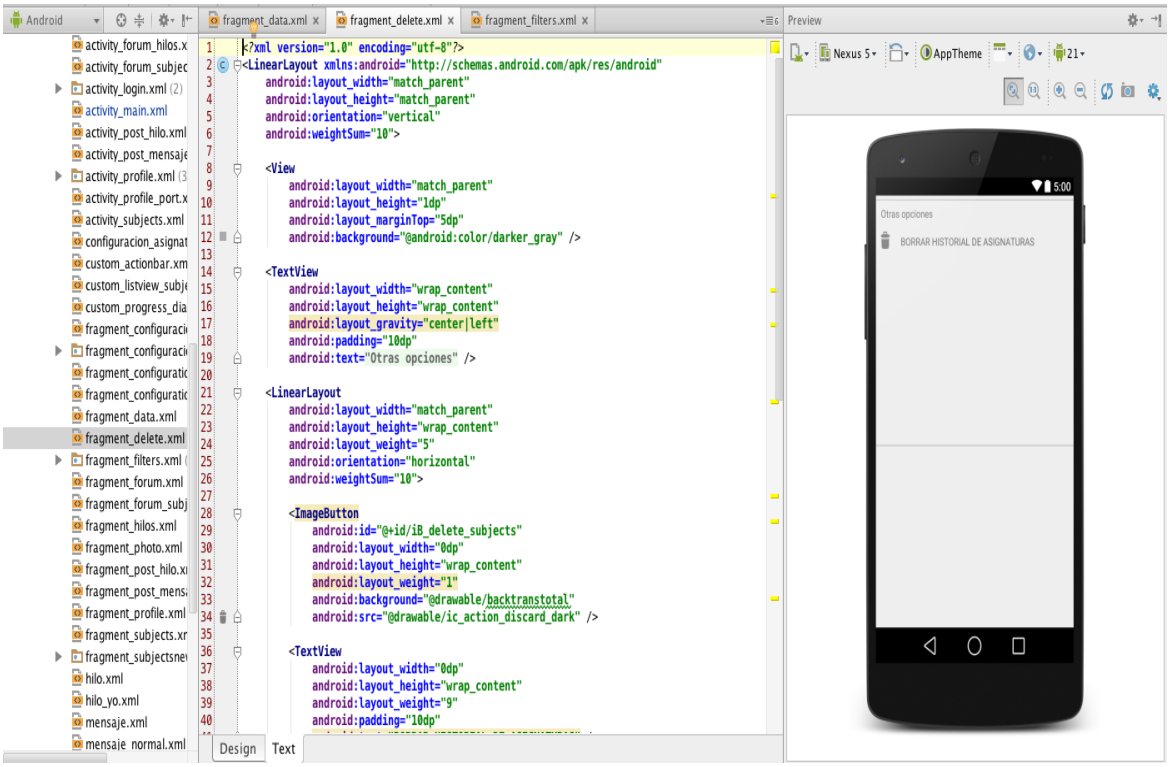


Fig. 3.52 Documento XML *fragment_delete* que define la interfaz gráfica de la modificación de historial

Modificación de asignaturas

La modificación de asignaturas está presente en la aplicación del profesor, permitiendo a este modificar la información de la asignatura, su título y su fotografía. Para ello, en la vista de usuario se listan las asignaturas del profesor, y una vez seleccionada la asignatura a modificar se pasa a otra vista, definida por el documento XML *fragment_configuración_asignaturas*, mostrado en la Figura 3.53. Aquí, el profesor se encontrará con una vista parecida a la de creación de asignaturas, en la que podrá cambiar los atributos de la asignatura, siempre manteniendo la referencias a la asignatura previa a su modificación. Los datos se respaldan en el servidor, modificando la base de datos.

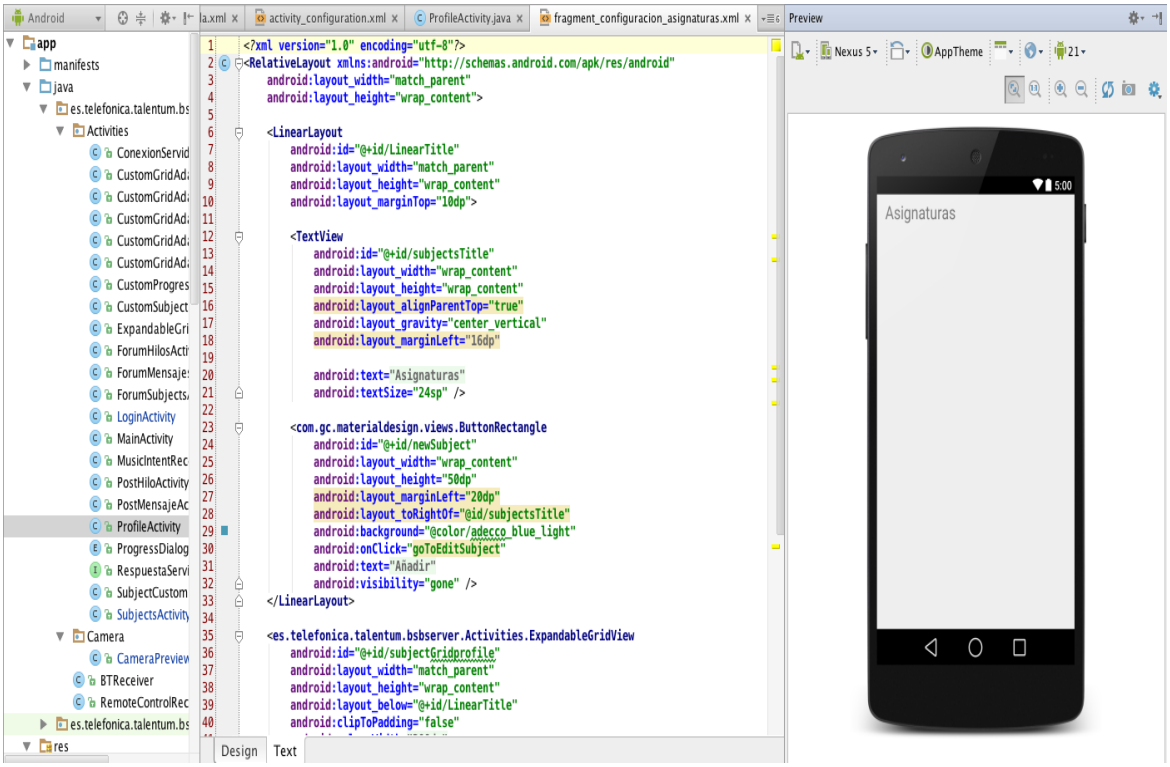


Fig. 3.53 Documento XML *fragment_configuración_asignaturas* que define la interfaz gráfica de la modificación de asignaturas

3.4 Pruebas de funcionamiento

Una vez terminado el análisis, diseño e implementación del proyecto, es hora de hacer una serie de test para comprobar el funcionamiento del sistema. Primero nos aseguraremos de

cumplir los requisitos anteriormente expuestos con unas pruebas de funcionalidad. Para terminar con unas pruebas de estrés, en las que se medirán el rendimiento del sistema en diferentes situaciones. Para definir las pruebas realizadas se volverá a utilizar el estándar IEEE 830 [32].

3.4.1 Pruebas de funcionalidad

El formato que tendrán las tablas será el seguido por la Tabla 3.35.

Nombre			**ID**
Descripción			
Acciones			
Erorres posibles			
Requisitos		Resultado	

Tabla 3.35 Tabla modelo prueba de funcionalidad

- **Nombre:** Nombre inequívoco de la prueba a modo de resumen de este.
- **ID:** Identificador inequívoco de la prueba. Estará formado por dos partes, el tipo de prueba y el código único de prueba.
- **Descripción:** Incluye una descripción breve de lo que incluirá la prueba.
- **Acciones:** Pasos a seguir para la realización de la prueba.
- **Errores posibles:** Errores que podrían surgir durante la prueba.
- **Requisitos:** Incluye una referencia a los requisitos involucrados en la prueba.
- **Resultado:** Según el resultado de la prueba será *Válido* o *Inválido*.

Entrada en el sistema			**PF-1**
Descripción	Comprobación de que los usuarios pueden entrar en el sistema a partir de la pantalla de login tras iniciar la aplicación.		
Acciones	1. Iniciar las apliaciones. 2. Introducir correo y contraseña. 3. Comprobar mensajes de error y login correcto de usuario.		
Erorres posibles	1. Contraseña mal introducida. 2. Email mal intoducido. 3. No hay conexión con el servidor.		
Requisitos	RFG-1	Resultado	*Válido*

Tabla 3.36 PF-1 Prueba entrada en el sistema

Vista principal			**PF-2**
Descripción	Comprobación del funcionamiento de las vistas principales de ambas aplicaciones. Mostrando en la aplicación del alumno la actividad reciente y en la del profesor sus asignaturas.		
Acciones	1. Entrar en el sistema. 2. Comprobar la existencia de actividad reciente relacionada con el usuario. 3. Comprobar las asignaturas que corrsponden al profesor.		
Erorres posibles	1. No se muestra ninguna información. 2. No se muestran los daots correctamemte. 3. Las imágenes no están actualizadas. 4. No hay conexión con el servidor.		
Requisitos	RFG-2, RFA-1, RFP-1	Resultado	*Válido*

Tabla 3.37 PF-2 Prueba vista principal

Cración de asignaturas			**PF-3**
Descripción	Comprobación del funcionamiento de la creación de una asignatura en el sistema para un profesor.		
Acciones	1. Entrar en el sistema. 2. Pulsar el botón de nueva asignatura. 3. Introuducir los datos y/o añadir una fotografía de la asignatura.		
Erorres posibles	1. No se registra la asignatura cerada en el sistema. 2. La imagen obtenida no es válida. 3. El texto introducudo no es válido.		
Requisitos	RFP-3	Resultado	*Válido*

Tabla 3.38 PF-3 Prueba creación de asignaturas

Cración de sesión de streaming			**PF-4**
Descripción	Comprobación del funcionamiento de la creación de una sesión de streaming desde la apliación del profesor.		
Acciones	1. Entrar en el sistema. 2. Seleccionar la asignatura de la que se quiere crear sesión. 3. Introuducir la configuración de privacidad y seguridad de la sesión.		
Erorres posibles	1. No aparecen las asignaturas del profesor. 2. La creción de la sala no queda registrada en el sistema. 3. La creción de la sala no tiene las características seleccionasdas.		
Requisitos	RFP-2, RFG-5	Resultado	*Válido*

Tabla 3.39 PF-4 Prueba creación de sesión de streaming

Reproducción de una sesión de streaming			**PF-5**
Descripción	Comprobación del funcionamiento de la unión a una sesión de streaming desde la apliación del alumno.		
Acciones	1. Entrar en el sistema. 2. Seleccionar una sesión o buscarla e introducir los credenciales si es necesario. 3. Comprobar la recepción de vídeo. 4. Comprobar la recepción de audio 5. Comprobar el funcionamiento de las notas. 6. Comprobar el funcionamiento de las preguntas. 7. Comprobar el funcionamiento de las capturas de pantalla.		
Erorres posibles	1. No aparecen las sesiones en curso. 2. La seguridad de las sesiones no es la correcta. 3. No se establece la conexión por WebSockets. 4. No se recepciona el vídeo. 5. No se recepciona el audio 6. Las notas no se almacenan. 7. Las preguntas no llegan al profesor. 8. Las capturas de pantalla no se guardan.		
Requisitos	RFA-2, RFA-3, RFA-4, RFA-5, RFA-6, RFA-7, RFG-5	Resultado	*Válido*

Tabla 3.40 PF-5 Prueba reproducción de una sesión de streaming

Creación de contenido para el repositorio			**PF-6**
Descripción	Comprobación del funcionamiento del almacenamiento de infromación creada durante una sesión de streaming a la vista de repositorio.		
Acciones	1. Entrar en el sistema. 2. Seleccionar una sesión o buscarla e introducir los credenciales si es necesario. 3. Creación de notas. 4. Creación de capturas de pantalla.		
Erorres posibles	1. La transcripción del profesor no se almacena. 2. Las notas no se almacenan. 3. Las capturas de pantalla no se guardan.		
Requisitos	RFA-8	Resultado	*Válido*

Tabla 3.41 PF-6 Prueba creación de contenido para el repositorio

Realizar conversación por el foro			**PF-7**
Descripción	Comprobación del funcionamiento del foro.		
Acciones	1. Entrar en el sistema. 2. Participar en una sesión de streaming. 3. Seleccionar una asigantura. 4. Seleccionar un hilo o crearlo. 5. Introducir un mensaje, responderlo o modificar uno tuyo.		
Erorres posibles	1. No se muestran las asignaturas en las que se ha participado al menos una vez en alguna sesión. 2. No aparecen los hilos correspondientes a una asignatura. 3. No aparecen los mensajes correspondientes a un hilo. 4. No se permite la creación de un hilo o mensaje. 5. No se permite la edición de un hilo o mensaje. 6. La estructura de los hilos y mensajes no se muestra correctamente.		
Requisitos	RFG-4	Resultado	*Válido*

Tabla 3.42 PF-7 Prueba realizar conversación por el foro

Modificar datos de usuario			**PF-8**
Descripción	Comprobación del funcionamiento de la vista de perfil y las modificaciones de usuario. Además en la aplicación del alumno se probarán diferentes configuraciones, mientras que en la aplicación del profesor se probará la modificación de asignaturas.		
Acciones	1. Entrar en el sistema. 2. Abrir la vista de perfil. 3. Comprobar el funcionamiento del cambio de contraseña, nombre de usuario y foto de perfil. 4. Comprobar el funcionamiento de las configuraciones de la aplicación del alumno. 5. Comprobar el funcionamiento de la modificación de asignaturas de la aplicación del profesor. 5. Comprobar el funcionamiento del borrado de historial.		
Erorres posibles	1. Los datos de usuario mostrados no son los correctos. 2. Los cambios de usuario no se mantienen. 3. La constraseña introducida no es válida. 4. La foto capturada no es válida. 5. El borrado de historial no se registra en el sistema. 6. El cambio de configuración no funciona. 7. La modificación de asignaturas no es correcta.		
Requisitos	RFG-3, RFG-7, RFA-9, RFP-4	Resultado	*Válido*

Tabla 3.43 PF-8 Prueba modificar datos de usuario

3.4.2 Pruebas de estrés

El formato que tendrán las tablas será el seguido por la Tabla 3.44.

Nombre	**ID**
Descripción	
Resultado	

Tabla 3.44 Tabla modelo prueba de estrés

- **Nombre:** Nombre inequívoco de la prueba a modo de resumen de este.
- **ID:** Identificador inequívoco de la prueba. Estará formado por dos partes, el tipo de prueba y el código único de prueba.
- **Descripción:** Incluye una descripción breve de lo que incluirá la prueba.
- **Resultado:** Se resumen los resultados obtenidos en la realización de la prueba.

Velocidad de carga vista principal	**PE-1**
Descripción	Medición de tiempos de respuesta y bytes descargados durante el listado de actividad reciente en la aplicación del alumno.
Resultado	Conexión de WiFi a 65 Mbps a 2.4 Ghz: 1. 5 elementos listados, 1,67 Mbytes en 2,250 segundos. 2. 10 elementos listados, 2,84 Mbytes en 3,750 segundos. 3. 15 elementos listados, 5,19 Mbytes en 6,550 segundos. Conexión de WiFi a 150 Mbps a 5 Ghz: 1. 5 elementos listados, 1,67 Mbytes en 0,750 segundos. 2. 10 elementos listados, 2,84 Mbytes en 1,250 segundos. 3. 15 elementos listados, 5,19 Mbytes en 1,750 segundos.

Tabla 3.45 PE-1 Velocidad de carga de la vista principal

Velocidad de subida de imágenes al servidor	**PE-2**
Descripción	Medición de tiempos de respuesta y bytes subidos durante la toma de una captura de pantalla durante una sesión de streaming en la aplicación del alumno.
Resultado	Conexión de WiFi a 65 Mbps a 2.4 Ghz: 0,25 Mbytes en 1,250 segundos. Conexión de WiFi a 150 Mbps a 5 Ghz: 0,25 Mbytes en 0,500 segundos.

Tabla 3.46 PE-2 Velocidad de subida de imágenes al servidor

Transmisión de audio y vídeo durante el streaming	**PE-3**
Descripción	Medición del ancho de banda utilizado y delay durante una sesión de streaming.
Resultado	Conexión de WiFi a 65 Mbps a 2.4 Ghz: 1. Calidad baja, 293,1 Kbytes de ancho de banda y 0,300 segundos de delay. 2. Calidad media, 488,3 Kbytes de ancho de banda y 0,500 segundos de delay. 3. Calidad alta, 683,6 Kbytes de ancho de banda y 1,200 segundos de delay. Conexión de WiFi a 150 Mbps a 5 Ghz: 1. Calidad baja, 293,1 Kbytes de ancho de banda y 0,100 segundos de delay. 2. Calidad media, 488,3 Kbytes de ancho de banda y 0,300 segundos de delay. 3. Calidad alta, 683,6 Kbytes de ancho de banda y 0,800 segundos de delay.

Tabla 3.47 PE-3 Transmisión de audio y vídeo durante el streaming

Capítulo 4

Planificación y presupuesto

En este capítulo se realiza un detalle de la planificación y presupuesto del sistema implementado en este proyecto. Primero definiremos la metodología de trabajo empleada, para después desarrollar una planificación de tareas que vendrá acompañada de un diagrama de Gantt, a modo de resumen gráfico. Por último se incluye un desglose de los costes del desarrollo e implementación del sistema de este proyecto.

4.1 Metodología de trabajo

En esta sección se detallarán las diferentes metodologías de trabajo seguidas a lo largo del desarrollo del proyecto. Además, al tratarse de un proyecto realizado dentro de un equipo, se explicarán las herramientas colaborativas utilizadas.

4.1.1 Metodología de desarrollo

A la hora de desarrollar un proyecto de software existen diversas metodologías de desarrollo. Un proyecto de software [36] es una colección de actividades de trabajo y tareas a realizar un determinado producto. La estructura que sigue un modelo de software se define en cinco actividades principales: comunicación, planificación, modelado, construcción y despliegue. A continuación, explicaremos los modelos de software más relevantes en la ingeniería del software, para después matizar el modelo seguido en el desarrollo de este proyecto.

Desarrollo en cascada

El modelo de cascada es denominado así debido a la posición de las fases de desarrollo, que caen como una cascada al empezar cada fase, siempre que la fase anterior haya finalizado. Este modelo tiene un enfoque sistemático y secuencial en el que las actividades anteriormente citadas se suceden de forma lineal. El modelo es ideal para situaciones en la que los requerimientos son fijos. Sin embargo este modelo presenta varias limitaciones, ya que a la hora de realizar un proyecto real, los requisitos de cliente van modificando o aumentando, y adaptar un cambio de este tipo supone un replanteamiento entero del modelo. Un diagrama del modelo de desarrollo en cascada es mostrado en la Figura 4.1.

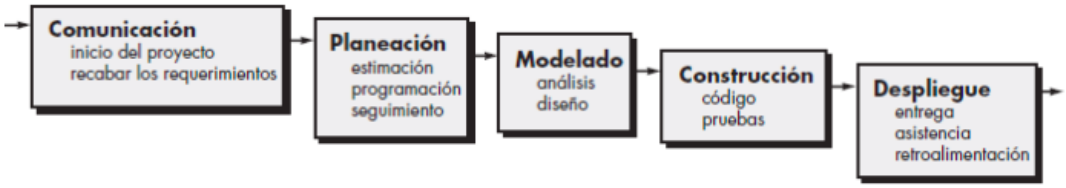


Fig. 4.1 Modelo de desarrollo en cascada [36]

Desarrollo iterativo y creciente

Este modelo de desarrollo se caracteriza por ser un conjunto de tareas agrupadas en pequeñas etapas repetitivas, conformando cada iteración. Se denomina también creciente porque las funcionalidades del proyecto van aumentando en cada iteración. El modelo iterativo se adapta a un modelo de proyecto en el que los requerimientos del producto cambian según avanza el desarrollo del producto, cuando el cliente define unos objetivos principales al inicio, pero a lo largo del desarrollo del producto se van detallando. Este modelo es muy popular debido a que está ligado a muchas estrategias de desarrollo de software y programación extrema, como es el desarrollo ágil, en el que en cada iteración se desarrolla un prototipo funcional al que se le van añadiendo nuevas características. Por contra, presenta la dificultad de controlar el avance de los proyectos en cada una de sus etapas. Un diagrama del modelo de desarrollo iterativo y creciente es mostrado en la Figura 4.2.

En conclusión, tras definir diferentes modelos de desarrollo de software, se decide basar nuestra metodología en un desarrollo iterativo y creciente, muy cercano al uso de técnicas de desarrollo ágil del software. La elección de este modelo de desarrollo se basa en que cada etapa iterativa se implementará un módulo más de la aplicación que irá mejorando las

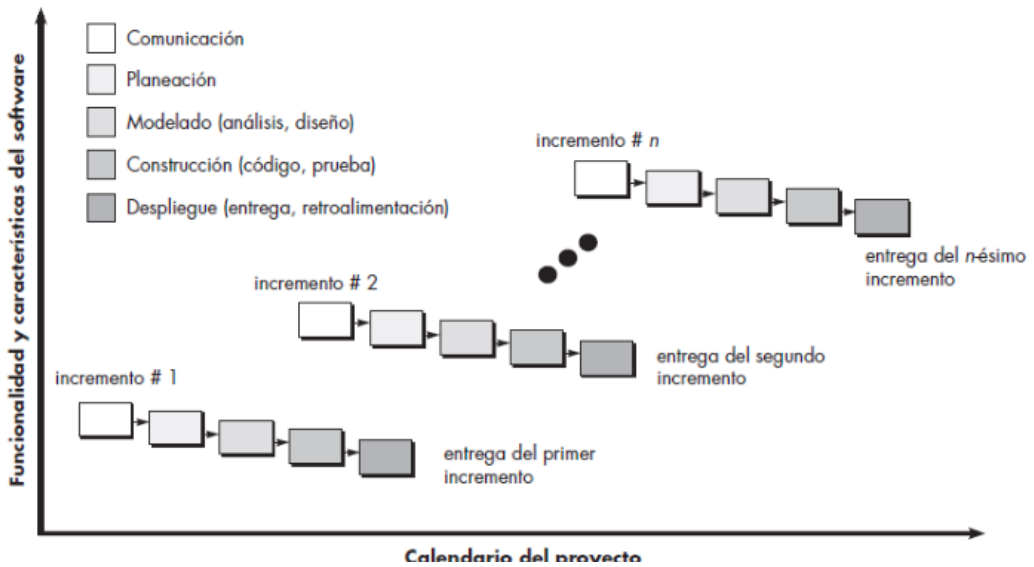


Fig. 4.2 Modelo de desarrollo iterativo y creciente [36]

funcionalidades de los módulos anteriores. Además, el desarrollo de algunas funcionalidades se podrá desarrollar en paralelo para agilizar la implementación conjunta de los módulos. Un modelo en cascada no favorecería el desarrollo de este proyecto, ya que las funcionalidades van aumentando a lo largo del desarrollo del proyecto en las diferentes etapas para los diferentes módulos, sin seguir una estructura lineal. Otra desventaja sería no poder contar con prototipos funcionales hasta el final del desarrollo del proyecto.

4.1.2 Uso de herramientas colaborativas

El hecho de que este proyecto se haya desarrollado dentro de un equipo, ha ido necesario el uso de herramientas colaborativas que permitiesen la coordinación sin intromisión.

La primera herramienta necesaria fue la de crear un repositorio para el código fuente. Se crearon tres diferentes repositorios, uno para cada aplicación Android y otro para el servidor. El sistema de control de versiones utilizado fue Git, a través de la plataforma BitBucket, donde se alojaría el código remotamente de una manera gratuita, privada y con capacidad de hasta cinco usuarios. Se utilizó SourceTree como herramienta visual para la gestión de los repositorios, además de las integraciones que ofrecen los entornos de desarrollo con repositorios Git, dando funcionalidad básica. Son muchas las ventajas de utilizar este tipo de sistemas de control de versiones. Estos permiten volver a versiones anteriores del código, por

si algún fallo surgiese. También permite trabajar a varias personas a la vez sobre un mismo código mediante la utilización de ramas. Con este último mecanismo, Git permite trabajar a varias personas sobre una misma funcionalidad, y tras finalizar la tarea, unir el código de ambos, uniendo las ramas, y resolviendo conflictos entre partes del código modificadas por ambas personas de cara a una resolución final. Estas herramientas se muestran en la Figura 4.3.

Fig. 4.3 Herramientas de control de versión

Para la comunicación entre los integrantes del equipo fue de gran ayuda la utilización de Trello, una aplicación web y móvil cuya función es ayudar a la organización de proyectos, sirviendo como tablón de ideas y lista de tareas. Por otra parte, para el intercambio de archivos, se optó por el uso de una carpeta compartida en el servicio Google Drive. Su integración dentro de los escritorios en los diferentes sistemas operativos, en la que se tiene una carpeta en el propio disco duro localmente sincronizada, hizo que la trasferencia de archivos fuese rápida y sencilla. Estas herramientas se muestran en la Figura 4.4.

Fig. 4.4 Herramientas de colaboración

Por último cabe mencionar que se utilizó, aunque no para colaborar entre el equipo, y más como ayuda para desarrollar código y obtener ejemplos, las plataformas GitHub, Stack Overflow y Google Developers. Estas plataformas se muestran en la Figura 4.5.

Fig. 4.5 Herramientas de desarrollo

4.2 Planificación

Una vez definido el modelo de desarrollo que vamos a seguir en la realización del proyecto, vamos a definir las actividades que conformarán nuestro modelo y las fases en las que se desarrollará.

Cada fase estará compuesta por un previo análisis y diseño del módulo en cuestión. Tras esto vendrán la implementación de las diferentes funcionalidades y una fase de pruebas, antes de poder pasar al siguiente módulo. La independencia entre algunas funcionalidades a permitido en ciertas ocasiones desarrollos en paralelo. Las fases del desarrollo cuentan con la implementación en ambas aplicaciones y servidor.

Las fases de desarrollo del proyecto son las siguientes:

1. **Desarrollo módulo inicial:** Este primer módulo consta de un intenso análisis y diseño del proyecto a realizar, estableciendo las ideas base y definiendo el alcance de este. Se establecerán los entorno de trabajo y el primer prototipo de aplicación.
2. **Desarrollo módulo comunicaciones:** A lo largo de este módulo se hará el diseño de la arquitectura de comunicaciones en la que se basarán los futuros módulos. Además se diseñará e implementará la base de datos.
3. **Desarrollo módulo vista login:** Este módulo añadirá a la aplicación una primera vista para la identificación de usuarios que entren en el sistema. Se hará especial hincapié en la seguridad de esta funcionalidad.
4. **Desarrollo módulo vista principal:** En este módulo se creará la primera vista tras hacer login el usuario. En la aplicación del profesor se implementará la creación de asignaturas, mientras que en la aplicación del alumno se implementará la lista de actividad reciente y las tarjetas informativas.
5. **Desarrollo módulo vista streaming:** Esta fase es la principal del proyecto, ya que engloba las principales funcionalidades del sistema necesarias para cumplir los objetivos.

Además, esta fase es la más larga del proyecto debido a la complejidad de establecer la comunicación de vídeo y audio entre las aplicaciones.

6. **Desarrollo módulo vista repositorio:** Este módulo se encargará de la implementación de la vista de repositorio, recuperando los datos generados por las vista de streaming y ofreciendo datos a la vista principal.

7. **Desarrollo módulo vista foro:** Esta fase se encarga de desarrollar el foro de las aplicaciones. Además se implementará la sincronización con la vista de streaming.

8. **Desarrollo módulo vista configuración:** Este módulo recoge los datos del usuario, permitiendo su modificación. También se implementarán funcionalidades para el manejo de diferentes configuraciones de la propia aplicación.

9. **Desarrollo de trabajo de fin de grado:** Esta última fase del proyecto dedicada la documentación de la memoria y la preparación de la defensa ante el tribunal.

En las siguientes figuras, Figura 4.6, Figura 4.7 y Figura 4.8. podemos observar el diagrama de Gantt de la planificación seguido, además del reparto de tiempos por fase y tarea. Durante el desarrollo del proyecto se han trabajado durante 25 horas semanales repartidas en 5 días, mientras que durante el desarrollo de la memoria del proyecto se han dedicado 20 horas semanales en 5 días. En la planificación se excluyen los días festivos y los correspondientes a fechas de exámenes, donde no se ha progresado en el proyecto. El desarrollo del proyecto ha tenido lugar en 6 meses, entre Noviembre de 2014 y Mayo de 2015, con un total de 111 días trabajados y 555 horas dedicadas. Por otro lado, el desarrollo de la documentación de la memoria ha durado 3 meses, entre Julio de 2015 y Septiembre de 2015, con un total de 70 días trabajados y 280 horas empleadas.

En resumen, el desarrollo total del proyecto ha sido llevado a cabo en 9 meses, de los cuales 181 días trabajados y un total 835 horas.

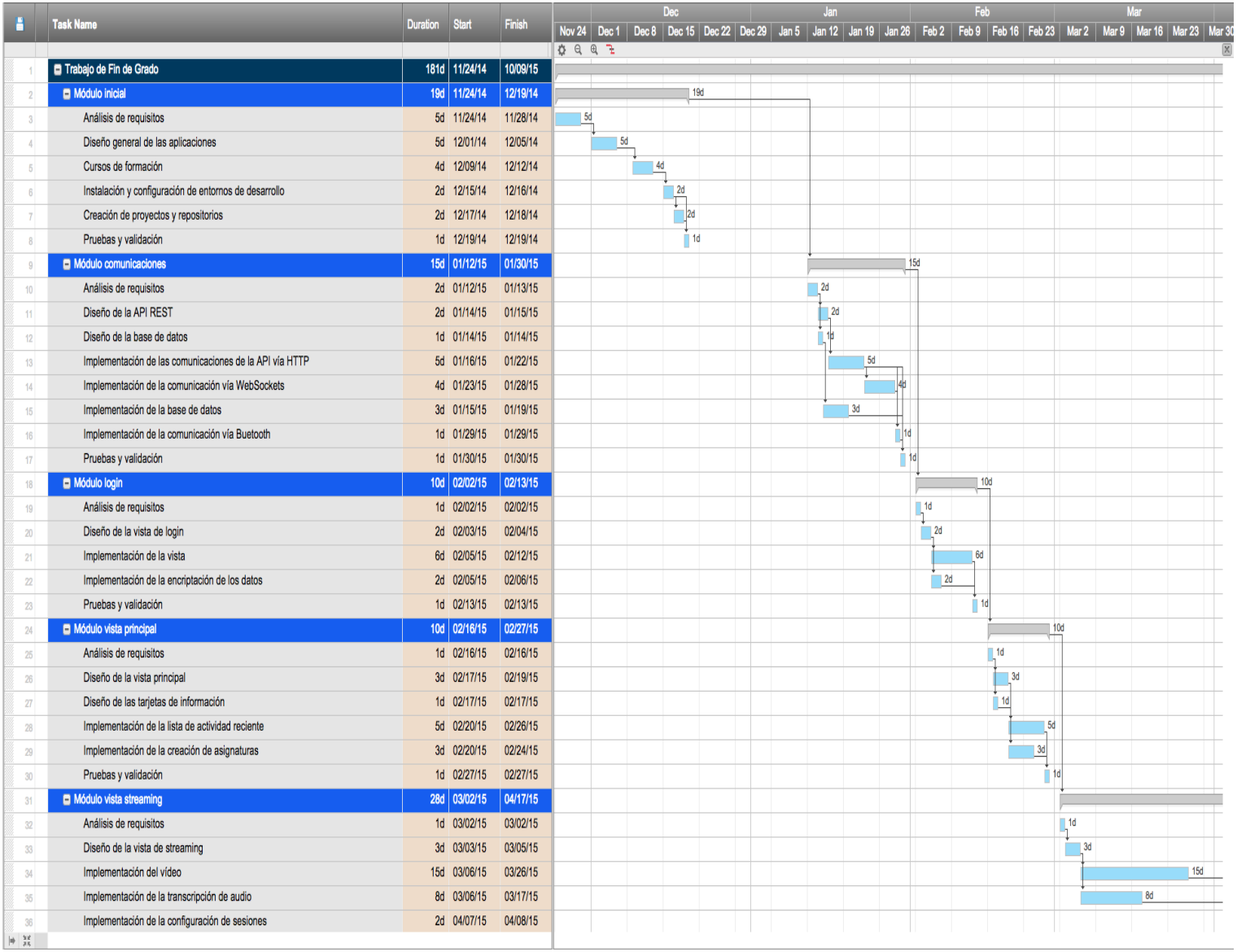
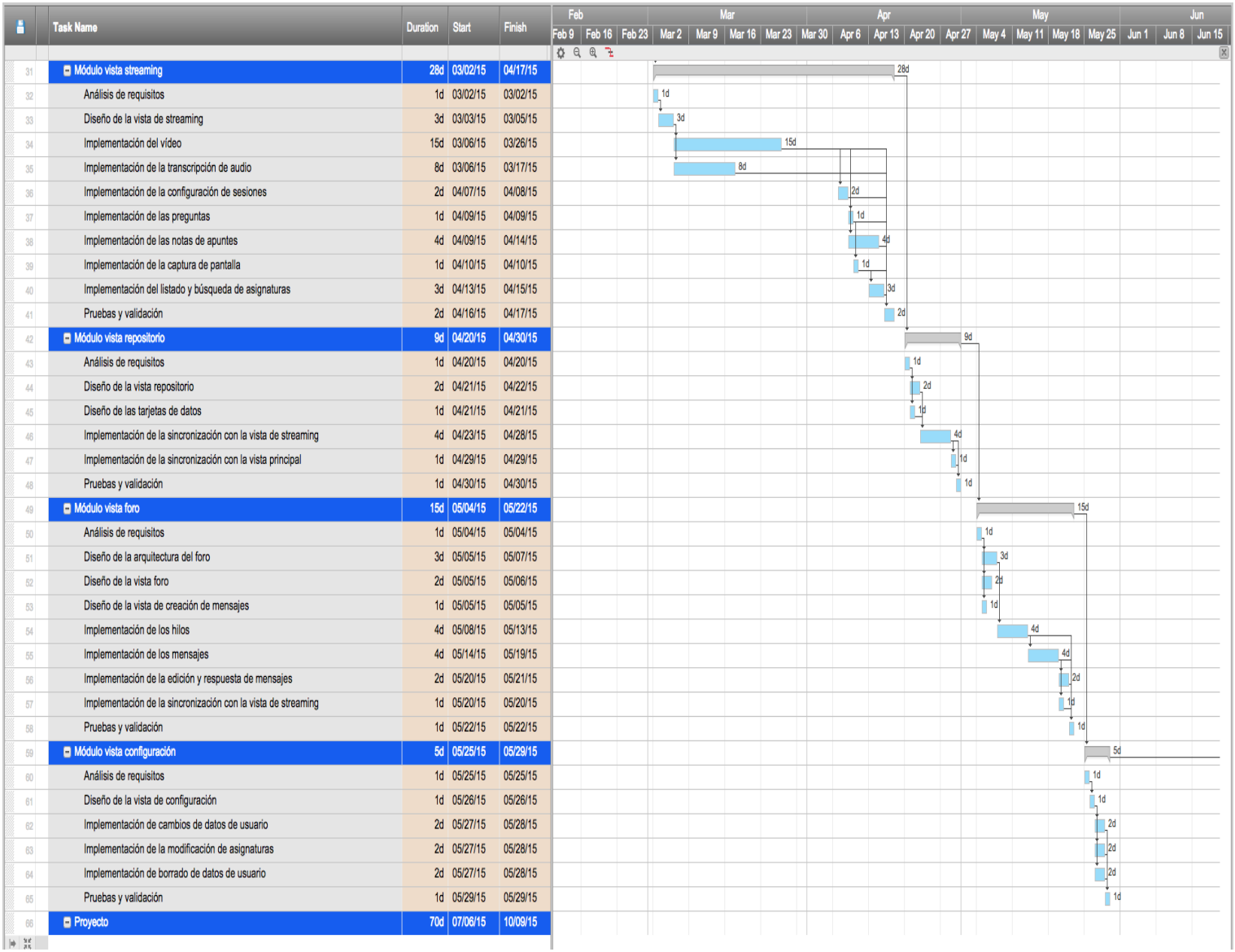


Fig. 4.6 Primer tercio del diagrama de Gantt y la planificación de actividades



	Task Name	Duration	Start	Finish
31	Módulo vista streaming	28d	03/02/15	04/17/15
32	Análisis de requisitos	1d	03/02/15	03/02/15
33	Diseño de la vista de streaming	3d	03/03/15	03/05/15
34	Implementación del vídeo	15d	03/06/15	03/26/15
35	Implementación de la transcripción de audio	8d	03/06/15	03/17/15
36	Implementación de la configuración de sesiones	2d	04/07/15	04/08/15
37	Implementación de las preguntas	1d	04/09/15	04/09/15
38	Implementación de las notas de apuntes	4d	04/09/15	04/14/15
39	Implementación de la captura de pantalla	1d	04/10/15	04/10/15
40	Implementación del listado y búsqueda de asignaturas	3d	04/13/15	04/15/15
41	Pruebas y validación	2d	04/16/15	04/17/15
42	Módulo vista repositorio	9d	04/20/15	04/30/15
43	Análisis de requisitos	1d	04/20/15	04/20/15
44	Diseño de la vista repositorio	2d	04/21/15	04/22/15
45	Diseño de las tarjetas de datos	1d	04/21/15	04/21/15
46	Implementación de la sincronización con la vista de streaming	4d	04/23/15	04/28/15
47	Implementación de la sincronización con la vista principal	1d	04/29/15	04/29/15
48	Pruebas y validación	1d	04/30/15	04/30/15
49	Módulo vista foro	15d	05/04/15	05/22/15
50	Análisis de requisitos	1d	05/04/15	05/04/15
51	Diseño de la arquitectura del foro	3d	05/05/15	05/07/15
52	Diseño de la vista foro	2d	05/05/15	05/06/15
53	Diseño de la vista de creación de mensajes	1d	05/05/15	05/05/15
54	Implementación de los hilos	4d	05/08/15	05/13/15
55	Implementación de los mensajes	4d	05/14/15	05/19/15
56	Implementación de la edición y respuesta de mensajes	2d	05/20/15	05/21/15
57	Implementación de la sincronización con la vista de streaming	1d	05/20/15	05/20/15
58	Pruebas y validación	1d	05/22/15	05/22/15
59	Módulo vista configuración	5d	05/25/15	05/29/15
60	Análisis de requisitos	1d	05/25/15	05/25/15
61	Diseño de la vista de configuración	1d	05/26/15	05/26/15
62	Implementación de cambios de datos de usuario	2d	05/27/15	05/28/15
63	Implementación de la modificación de asignaturas	2d	05/27/15	05/28/15
64	Implementación de borrado de datos de usuario	2d	05/27/15	05/28/15
65	Pruebas y validación	1d	05/29/15	05/29/15
66	Proyecto	70d	07/06/15	10/09/15

Fig. 4.7 Segundo tercio del diagrama de Gantt y la planificación de actividades

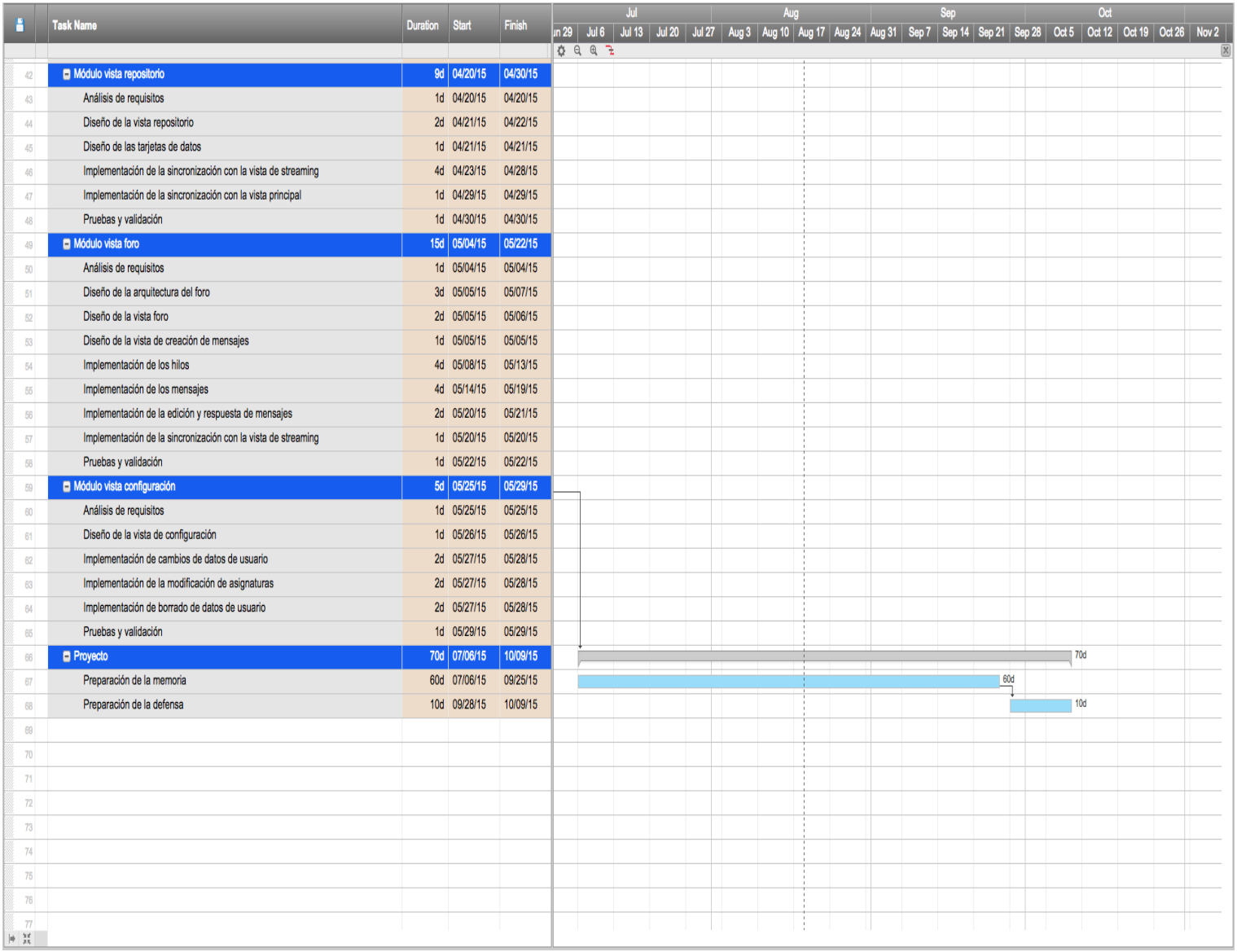


Fig. 4.8 Tercer tercio del diagrama de Gantt y la planificación de actividades

4.3 Presupuesto

Durante esta sección ofreceremos en detalle los costos que han conllevado el desarrollo de este proyecto. Al tratarse de un proyecto realizado en empresa, se explican por separado los costes del desarrollo del proyecto de los costes de implementación de la solución.

4.3.1 Desarrollo del proyecto

El desarrollo de este proyecto ha sido llevado a cabo dentro de un equipo compuesto por un ingeniero junior y un ingeniero senior, director del proyecto. Para el cálculo del presupuesto del desarrollo del proyecto se asume sólo el coste de personal, las nóminas netas de los integrantes del equipo, ya que se desconocen los costes llevados por la empresa en materia de instalaciones, seguridad social, etc. En la Tabla 4.1 se muestran los costes de personal.

Personal	Categoria	Coste hombre/mes (Euros)	Coste(Euros)
Diego Domínguez Alvarez	Ingeniero Junior	500	3000
Enrique Mendoza Robaina	Ingeniero Senior	500	3000
Coste total		1000	6000

Tabla 4.1 Coste de personal

4.3.2 Implementación

Para la implementación de la solución propuesta se tendrá en cuenta el equipo utilizado para la consecución del proyecto. El coste de equipo se calcula a partir de una amortización debido a los meses utilizados durante el desarrollo del proyecto. Se establece una amortización de 1/4 para el ordenador y de 1/6 para los diferentes dispositivos, ya que un ordenador tiene un valor de depreciación mayor en el tiempo. En la Tabla 4.2 se muestran los costes de implementación.

Descripción	Precio(Euros)	Amortización	Coste(Euros)
Ordenador MacBook Air	1100	1/4	275,00
Dispositivo móvil Google Nexus 5	350	1/6	58,33
Dispositivo tableta Google Nexus 7	220	1/6	36,67
Dispositivo bluetooth BeeWi BBH100	50	1/6	8,33
Coste total			378,33

Tabla 4.2 Coste de implementación

4.3.3 Coste total

Con esto se puede concluir que el coste total del proyecto asciende a la cantidad de **seis mil trescientos setenta y ocho con treinta y tres céntimos de euros**. En la Tabla 4.3 se muestran los costes de implementación.

Descripción	Coste(Euros)
Coste desarrollo proyecto	6000,00
Coste implementación proyecto	378,33
Coste total	6378,33

Tabla 4.3 Coste total

Chapter 5

Conclusions and future work

In this chapter we will make some conclusion about the work done in this project after finishing it. We will also propose some future work that can be applied to improve it.

5.1 Conclusions

In this section we are going to expose the conclusions that we found from two perspectives. First we are going to talk about the project conclusions, and after that, the personal conclusions we have obtained after finishing this project.

5.1.1 Project conclusions

At this point, we can say that we have satisfied our main objective, which was to create a system of applications and a server that together can solve the problem of divided attention that deaf students have when they are in class at the university.

Without any doubt, we have accomplished our primary goals:

- It has been developed an app for the professor that is capable of recording the audio and video of the class, doing a streaming to their students. This app has an easy

management of the different courses a professor can have and a fast set up of the streaming sessions.

- It has been developed an app for the students that is able to receive the streaming of audio and video from the professor. As well as taking notes, asking questions and making screenshots at the same time of the streaming session.
- A server has been implemented to manage the connections between the professor and the students. Also the server is able to manage the system and the information stored in the database.

Together to the primary goals, we can also state that the secondary goals have been achieved:

- The development of the project has used recent technologies that are open source such as WebSockets, the Android platform, MySQL and many more.
- We have built an integrative system, which not only helps deaf students to follow the classes, it is also a tool for any student that makes easier to pay attention to the class.
- The professor app has an easy access to set up sessions and get a class up and running in few steps.
- The student app is developed with many tools to the student. It has the ability to take notes, ask questions and take screenshots during a streaming session. As well as a repository and a forum to keep with the study after class.
- The solution can be easily being deployed in a university environment due to the fact that requirements of the system are not difficult to achieve by any university. That is getting the server and the apps running, we just need a few devices.

All these objectives have been accomplished because from the beginning the requisites where fully stated in a clear way and the development was following the planning from the start to the end of the project. Apart from them, many other functionalities have been added during the development of this project in order to increase the performance of the overall system.

Taking into account all this conclusions, we can state that all the objectives of this project have been achieved successfully.

5.1.2 Personal conclusions

From the first time, when I obtained the scholarship by Talentum I faced this project as a personal challenge. It was the first time I was working for a company to develop a project within a time range, the time it lasted the scholarship. Here also starts my decision to make this project the one to be developed in my bachelor's thesis.

I faced this challenge from the beginning as a great chance to improve my personal skills, technically and interpersonally. I didn't have any previous experience building Android applications, servers or databases, but I had strong knowledge in programming. This project has allowed me to expand my knowledge in programming skills to another level. Also working inside a team, has make me improve my qualities to work in a team and under pressure, due to the fact it was an extensive project with a due date.

As I have finished this project, I can conclude that the development of this project, together with the years of the degree, has made me increase the interest in the new technologies and the passion to engineering, giving me the felling that I can face any problem that comes, opening me new perspectives and paths to the future.

5.2 Future work

This aim of this section is to propose some future lines that can be followed to improve the present project. Although the project is finished, this propositions, if implemented, will make this project more complete and powerful to complete the tasks it was developed for. Below are listed some ideas that can be applied to this project:

5.2.1 Web Administration

One of the disadvantages that this protect has, is a unified place for managing the system. At the moment, managing configurations, data of users or managing new accounts have to be done going straight to the code or through database managers. For that, it is proposed to create a web page for administration purposes.

This tool will be accessed from anywhere of the internet. It will have a login process for the administrator allowed from each institution, to control the different systems. The main windows it will have are the following:

- **Users management:** This will allow the administrator to add new users to the system and manage their characteristics.
- **Data management:** Here it will be managed all the data stored in the server apart from an interface to manage the information stored in the database.
- **Network management:** This will allow to change the network parameters for the communication between the devices.
- **Statistics:** This window will gather statistics from users, network and other quantifiable information for future analysis.

5.2.2 Streaming

One of the main parts of this project that allows to accomplish the main objective of the system, is the streaming of video. Although, Motion JPEG accomplishes its task of streaming video, it is much work to be done in this part, to became a powerful real time streaming video tool.

For this purpose, it is proposed that in the future it will be needed to change the way the video is streamed. One of the first options, that didn't work in this project, is video coding. It could be investigated to implement new video coding techniques that will allow better quality of video and less delay. Other option could be to externalize this section to a streaming service. This could be done with external paid streaming service providers like Wowza servers [37].

5.2.3 Speech recognizer

Together with the video streaming, another important part of this system is the speech transcription. Google Speech Recognizer engine provides free, fast and accurate transcription. But sometimes lacks of precision to start to recognize the beginning of the speech, winch leads to miss the first word of some sentences.

Here it is proposed two solutions to be applied in the future. The first one is to implement our own speech recognizer, based on the one provided by Google, in order to improve the vulnerabilities, we have using their system. As this task seems a bit difficult to implement, it is also proposed to use an external paid service, like Dragon Naturally Speaking [38].

5.2.4 Deployment

The next future task to perform is to integrate the system developed in a real case scenario, that is deploying this project in a real University. The system is prepared for this scenario, but never was accomplished such a task. It is sure that some modifications will have to be made for a real deployment. This includes to set up correctly network settings, administration tools and many other details.

It will also be needed to buy the equipment necessary to operate the system developed. This is at least to have a computer to run the server, a mobile phone and a Bluetooth device for the teacher and a mobile device for the student. The characteristics of this devices are supposed to have at least the ones described in section 3.3.1. The cost of this implementation will be similar as the one proposed in section 4.3.2.

Depending on the policy we could agree on with the university, the applications will be uploaded to the Google Play Store. There we will access free to download the student app, while the teacher's app will be access free or paid, depending on the policy, with access key for each teacher to identify the different teachers from different universities. This part of the deployment has to be studied deeply for better results.

Referencias

[1] Licenses. https://source.android.com/source/licenses.html [En línea; acceso 28-Julio-2015].

[2] Android logo png. http://saqibsomal.com/2015/03/13/sony-vaio-debut-smartphone-officially-presented/android-logo-png-3/ [En línea; acceso 1-Septiembre-2015].

[3] Android (operating system). https://en.wikipedia.org/wiki/Android_(operating_system) [En línea; acceso 24-Julio-2015].

[4] ¿es android abierto? http://hipertextual.com/2015/02/es-android-abierto [En línea; acceso 1-Septiembre-2015].

[5] App stores growth accelerates in 2014. http://blog.appfigures.com/app-stores-growth-accelerates-in-2014/ [En línea; acceso 24-Julio-2015].

[6] Google i/o by the numbers: 1b android users, 900m on gmail. http://www.cnet.com/news/google-io-by-the-numbers-1b-android-users-900m-on-gmail/ [En línea; acceso 24-Julio-2015].

[7] Android version history. https://en.wikipedia.org/wiki/Android_version_history [En línea; acceso 24-Julio-2015].

[8] Dashboards. http://developer.android.com/intl/es/about/dashboards/index.html [En línea; acceso 1-Septiembre-2015].

[9] Supporting multiple screens. http://developer.android.com/intl/es/guide/practices/screens_support.html [En línea; acceso 1-Septiembre-2015].

[10] Reto Meier. *Professional Android 2 Application Development*. Wiley Publishing, Inc, 2010.

[11] Application fundamentals. http://developer.android.com/intl/ko/guide/components/fundamentals.html [En línea; acceso 25-Julio-2015].

[12] Activity. http://developer.android.com/intl/es/reference/android/app/Activity.html [En línea; acceso 1-Septiembre-2015].

[13] Art and dalvik. http://source.android.com/devices/tech/dalvik/ [En línea; acceso 1-Agosto-2015].

[14] Mª Celeste Campo Vázquez. *Apuntes de la asignatura Aplicaciones Móviles*. Universidad Carlos III de Madrid, 2015.

[15] Server (computing). https://en.wikipedia.org/wiki/Server_(computing) [En línea; acceso 3-Agosto-2015].

[16] Client–server model. https://en.wikipedia.org/wiki/Client%E2%80%93server_model [En línea; acceso 1-Septiembre-2015].

[17] Web application framework. http://docforge.com/wiki/Web_application_framework [En línea; acceso 3-Agosto-2015].

[18] Model–view–controller. https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller [En línea; acceso 5-Agosto-2015].

[19] Play framework. https://en.wikipedia.org/wiki/Play_framework [En línea; acceso 5-Agosto-2015].

[20] Relational database. https://en.wikipedia.org/wiki/Relational_database [En línea; acceso 5-Agosto-2015].

[21] Using php data objects. http://www.creativebloq.com/design/using-php-data-objects-1133026 [En línea; acceso 1-Septiembre-2015].

[22] Database. https://en.wikipedia.org/wiki/Database [En línea; acceso 5-Agosto-2015].

[23] Mysql. https://es.wikipedia.org/wiki/MySQL [En línea; acceso 5-Agosto-2015].

[24] Representational state transfer. https://es.wikipedia.org/wiki/Representational_State_Transfer [En línea; acceso 6-Agosto-2015].

[25] Inbound rest through the integration agent (gig part 8). https://support.xmatters.com/hc/en-us/articles/203674559-Inbound-REST-through-the-integration-agent-GIG-part-8- [En línea; acceso 1-Septiembre-2015].

[26] A beginner's guide to http and rest. http://code.tutsplus.com/tutorials/a-beginners-guide-to-http-and-rest--net-16340 [En línea; acceso 6-Agosto-2015].

[27] Websocket. https://en.wikipedia.org/wiki/WebSocket [En línea; acceso 6-Agosto-2015].

[28] The websocket protocol. https://tools.ietf.org/html/rfc6455 [En línea; acceso 6-Agosto-2015].

[29] The websocket api. https://w3c.github.io/websockets/#websocket [En línea; acceso 6-Agosto-2015].

[30] Websockets vs rest: Understanding the difference. http://www.pubnub.com/blog/websockets-vs-rest-api-understanding-the-difference/ [En línea; acceso 6-Agosto-2015].

[31] Introducing json. http://www.json.org/ [En línea; acceso 1-Septiembre-2015].

[32] IEEE. *Ieee std 830-1993, recommended practice for software requirements specifications.* Institute of Electronic and Electrical Engineers Press, 1993.

[33] A look at the basics of bluetooth technology. http://www.bluetooth.com/Pages/Basics.aspx [En línea; acceso 10-Agosto-2015].

[34] Modelo entidad-relación. https://es.wikipedia.org/wiki/Modelo_entidad-relaci%C3%B3n [En línea; acceso 8-Agosto-2015].

[35] Mjpeg (motion jpeg) video codec. http://www.digitalpreservation.gov/formats/fdd/fdd000063.shtml [En línea; acceso 20-Agosto-2015].

[36] Roger S. Pressman. *Ingeniería del software: un enfoque práctico. 7a ed.* McGraw-Hill, 2010.

[37] Wowza media streaming server and cloud solutions. http://www.wowza.com/ [En línea; acceso 1-Septiembre-2015].

[38] Dragon naturally speaking. http://www.nuance.com/dragon/index.htm [En línea; acceso 1-Septiembre-2015].

Appendix A

English summary

A.1 Chapter 1: Introduction

In this chapter we will present a general overview of the project, the motivation, its main goals, the socio-economic environment, regulatory framework and finally, we will present the organization of the whole document.

This project starts from an internship with Telefónica with the idea to create a technological solution to solve the problem of divided attention that deaf students face during their university classes. This solution is called Breaking Sound Barriers, and it is composed of two applications, an application for the professor, called BSB Server, an application for the student, called BSB App, and a server.

The motivation of developing this project is the need of solving the problem that impact many deaf students. They are not capable of following the class at the same time they write notes, attend to the interpreter or the blackboard.

That is why our main goal of this project is to create a system of applications and a server that together can solve the problem of divided attention that deaf students face when they are in class at university. Here also we can state some primary objectives: create an application for the teacher that is capable of recording the video as well as transcribing the audio, create another application for the student that will receive the video and the transcription of the audio at the same time it allows to take notes and ask questions, and a server that can manage the connections between the apps and manage the system and the data stored in it.

After that, we can analyze the socio-economic impact that this solution can have. This solution is focused to help people with hearing disabilities, but also to become a tool for everyone, that will allow to follow easier the classes at the universities. Also the aim of this project is to reduce the cost that universities expend in sign language interpreters, the actual solution to help deaf students.

As said before, this solution is aimed to reduce cost for universities, that is why from a regulatory point of view, the project is developed using open source software and technologies, to avoid licensing. All source code and libraries used in this project are under the Apache 2.0 license or the GNU version 2 license.

Finally, in this chapter we can see a detailed explanation of the structure of this document, where we are specifying what we are going to see in each chapter.

A.2 Chapter 2: State of the art

In this chapter we will see some technologies that are related to the project, in order to explain everything that is necessary for the understanding of the project. First, we will see what in the Android platform; a brief introduction, its history, architecture, main APIs (Application Programming Interfaces) and the development environment. Then we will see the technologies used in the server, such us the framework used, its database and the characteristics of the client-server connections.

Starting with the Android platform, where both applications will be developed, first we will see a brief introduction to what it is the platform and how its works, giving some data of the actual dimensions of this platform.

Secondly, we will see an explanation of the history of the Android platform from the first version, Cupcake, to the last, Lollipop, viewing some of the characteristics that each version brought to the operating system.

Thirdly, we will see the components of the Android architecture, which is built from five layers: application layer, application framework, libraries, Android Run Time and Linux Kernel.

From the point of view of the server, several technologies will be described. We will start defining what is a server and its types. Inside, we will explain a certain type of servers that are web applications frameworks, specifically the Play Framework, the one that is used for the development of this project.

After stating what is the server, we will talk about databases, defining what a database is and its types. We will center the explanation on relational databases, specially in MySQL the one that is used during the implementation of this project.

Finally, we will explore the technologies used for the client-server communication. Here we will describe what is a REST API, the definition of a WebSocket and the coding format JSON. All technologies used during the implementation of the server.

A.3 Chapter 3: Analysis, design and implementation of the system

In this chapter we will perform an analysis of the requirements of the project to the latter design the solution and finally explain how it it implemented. We will also make some test to measure the performance of the system.

In the analysis part, it will be described the requirements of the project. The client requirements are composed of two different types of requirements. The first one, functional requirements, will explain the features that the system needs to accomplish the purpose of this project. The second one, constraint requirements, will explain the restrictions that limit our system. All the requirements are divided in general requirements, server requirements, student app requirements and teacher app requirements.

In the design part, it will be described the design of the different parts of the system implemented in this project. First, it will be defined the system, that is defining the components of the system, which are the application for the teacher, the application for the student and the server.

Secondly, it will be explained the design of the architecture of the system. We will talk about how it is designed the API REST, how the WebSocket communication is being done and also how the Bluetooth connection behaves.

Thirdly, we will see how the database is designed. Here we will show a diagram of the whole database, where we can observe the relations between the tables and attributes. Also it will be included a description of the tables and attributes of the database.

Fourthly, it will be shown the design of the different views of the applications. Here we will see some screenshots of what the views of the applications will have. This views will be explained divided in modules, as the development of the project was done in modules. It will be described both professor and student applications.

Fifthly, we will talk about possible design alternatives that could be done instead, and why we did not implement this way. Between the alternatives proposed there is a change in the applications platform, the format of messages, type of database and different socket connection.

In the implementation part, we will see how the solution is implemented form a source code view. Firstly, we will explain the technological environment that allows us to develop the implementation of the project. To continue with a detailed explanation of the different parts of the project.

The project is implemented in several independent modules: communications module, login module, main view module, streaming module, forum module, repository module and user view module.

- **Communications module:** In this module it will be explained how it is implemented the API REST over the protocol HTTP, the communication via WebSockets between the Android apps and the server, the communication via Bluetooth between the Bluetooh device and the professor app, and the communication between the database and the server.
- **Login module:** Here it will be described how the login works and how the data is encrypted to provide security to the system.
- **Main view module:** This module shows how it is implemented the main view of both apps, explaining how the student app shows the recent activity of the system to the student, while the professor app shows the courses he has and how to create new ones.
- **Streaming module:** In this module it will be described how each streaming view is implemented in the two applications. It will also be explained how all the functionalities are implemented, such as how do you take notes, take screenshots or ask questions,

in the student app. While in the professor app it will be explained how a streaming session is configured.

- **Forum module:** This module explains how the forum of the system is implemented and its different views, such us the thread or message views.
- **Repository module:** Here it will be shown how it is implemented the repository that grabs the data generated during a streaming session, and how the view works.
- **User view module:** This module explains how the user management is developed. This module allows the user to change its personal information, delete the information in the system and modify other characteristics of the application.

Finally, in the testing part, it will be exposed some real case test that has been done to check and measure the functionalities of the system. Within this test, there have been done functional test, to check that the functionalities of the system are implemented and working. And also there have been done some stress test, to check the performance of the system in different environments.

A.4 Chapter 4: Budget and planning

In this chapter we will show the estimate cost of developing and implementing this project together with the organization that has been taken to develop this project. We will also cover the techniques used for collaborative work during this project.

We start this chapter explaining different working methodologies, like the waterfall model or the iterative and incremental model. To finally decide which model fits better for the development of our project, which is the iterative and incremental, due to the fact it shares characteristics with agile methodologies.

We also talk in this chapter about the collaborative tools used during the implementation of this project. For version control of the source code, it is used Git and Bitbucket as online repository.

After that, it will be detailed the plan followed to the development of this project, together with the preparation of the thesis. The development is divided in nine phases:

1. Development of initial module.
2. Development of communication module.
3. Development of login view module.
4. Development of main view module.
5. Development of streaming view module.
6. Development of repository view module.
7. Development of forum view module.
8. Development of configuration view module.
9. Development of thesis.

Finally, it will be described a detailed estimate of the cost of the project. First explaining the cost of development and then the cost of implementation. Together it sums up to six thousands three hundred seventy-eight with thirty-three cents euros.

A.5 Chapter 5: Conclusions and future work

In this chapter we will make some conclusion about the work done in this project and propose some future work that can be applied to improve it.

In the end, we make some conclusions about all the work done. Where we can observe in the project conclusions, that we have met all the objectives successfully, accomplishing all the requirements. Some personal conclusions are also explained.

To conclude this chapter and thesis, it is proposed some future work. Within this lines there is explained some improvements that can be implemented for a better performance of the system development. This future work includes this ideas: a web administration tool, improvement of the video streaming, improvement of the speech recognizer and a real case deployment in a university.

Anexo B

Manual de usuario

El objetivo de este anexo es describir en detalle como desenvolverse en las aplicaciones Android del proyecto, tanto en la aplicación del profesor como en la del alumno. Para que el usuario aprenda a navegar por la interfaz se seguirán una serie de pasos a modo de tutorial.

B.1 Aplicación profesor

B.1.1 Vista login

La aplicación empieza con la pantalla de login. Donde el usuario podrá introducir sus credenciales y tendrá la opción de recordarlos para un auto-login.

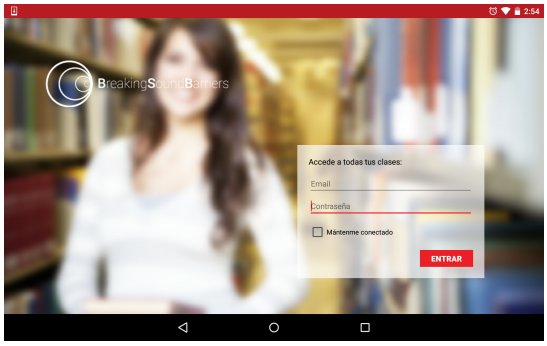


Fig. B.1 Vista login

B.1.2 Vista principal

Como profesor, la pantalla inicial muestra un listado de las asignaturas que ha creado ese profesor, más un botón para poder crear nuevas asignaturas. Desde la parte de arriba de la pantalla, la navigation bar, tenemos acceso directo a la vista de foro y a la vista de usuario. La pantalla principal que veremos será la siguiente.

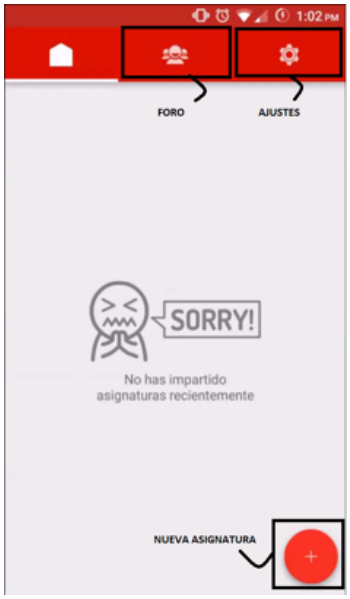


Fig. B.2 Vista principal profesor

B.1.3 Creación de asignaturas

Para poder añadir una asignatura, se debe pulsar el botón situado en la parte inferior derecha de la pantalla. Una vez se pulse el botón pasaremos a la creación de la asignatura de manera muy sencilla e intuitiva.

Se escribe el nombre de la asignatura que se va a impartir y, si se desea, una fotografía personalizada para la asignatura. Esta imagen puede ser tanto obtenida de la galería como directamente de una instantánea de la cámara.

Fig. B.3 Vista creación asignatura

B.1.4 Creación de sesiones

Tras configurar la asignatura que se va a impartir (esto sólo se hace la primera vez), se procede a la creación de la sala o sesión. Para ello, pulsamos el botón de la flecha siguiente dentro de la asignatura seleccionada.

Hay dos opciones para crear la sesión:

1. **Sesión privada:** Si sólo se desea que puedan entrar aquellas personas que están bajo la misma red WiFi, o lo que es lo mismo, que se encuentran dentro de la misma institución (universidad, empresa, etc.). Con esto se asegura la privacidad de la sesión.
2. **Acceso con contraseña:** Esta opción es para asegurar una mayor privacidad que la anterior, pues sólo podrán acceder aquellas personas que posean la clave de acceso. Esta contraseña será comunicada por el profesor a las personas que él desee (en general, a su clase).

Tras la configuración de privacidad de la sala, ya hemos terminado todo lo necesario para empezar una clase.

Una vez configurada la sesión a nuestro gusto, podemos abrir la sala, dando a dicho botón. Así entrando en la vista de streaming. Esta parte de la aplicación se puede dividir en tres partes:

1. **Para empezar la clase:** Pulsar el botón de play y se realizará la conexión con el resto de dispositivos que estén dentro de la sala.
2. **Durante la clase:** Con los cascos y el micrófono Bluetooth conectado, impartir la clase normalmente. Los auriculares Bluetooth conectados permiten escuchar las preguntas formuladas por los alumnos, ya que las preguntas serán pasadas a voz. Otra funcionalidad importante para la accesibilidad es la transcripción de voz a texto, para lo cual es necesario un micrófono Bluetooth conectado al dispositivo.
3. **Para finalizar la clase:** Pulsar el botón de stop que se mostrará justo cuando se haya iniciado la sesión o pulsar la flecha situada en la parte superior izquierda.

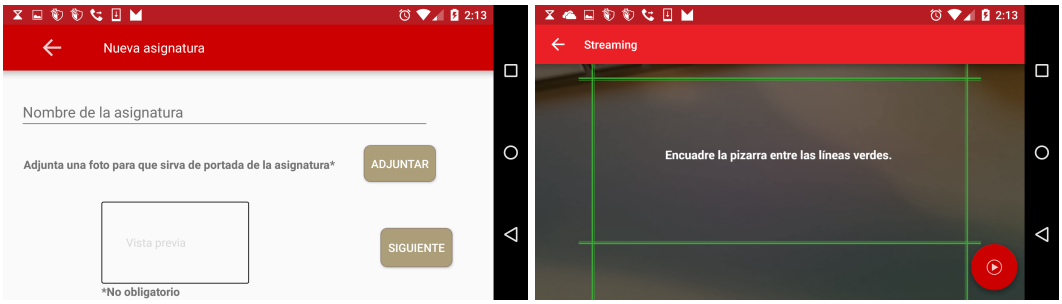


Fig. B.4 Vista streaming profesor

B.1.5 Foro

El foro se organiza por asignatura, que a su vez se agrupa por hilos, y dentro de estos, comentarios por hilo. En el foro aparecerán las asignaturas que imparte el profesor. Dentro de cada asignatura, aparecerán los diferentes temas de conversación. Aquí el usuario puede crear nuevos hilos o cambiar de nombre los existentes creados por él. Y dentro de cada hilo, el usuario podrá crear nuevos mensajes, contestar a otros que ya existan o editar los existentes creados por él.

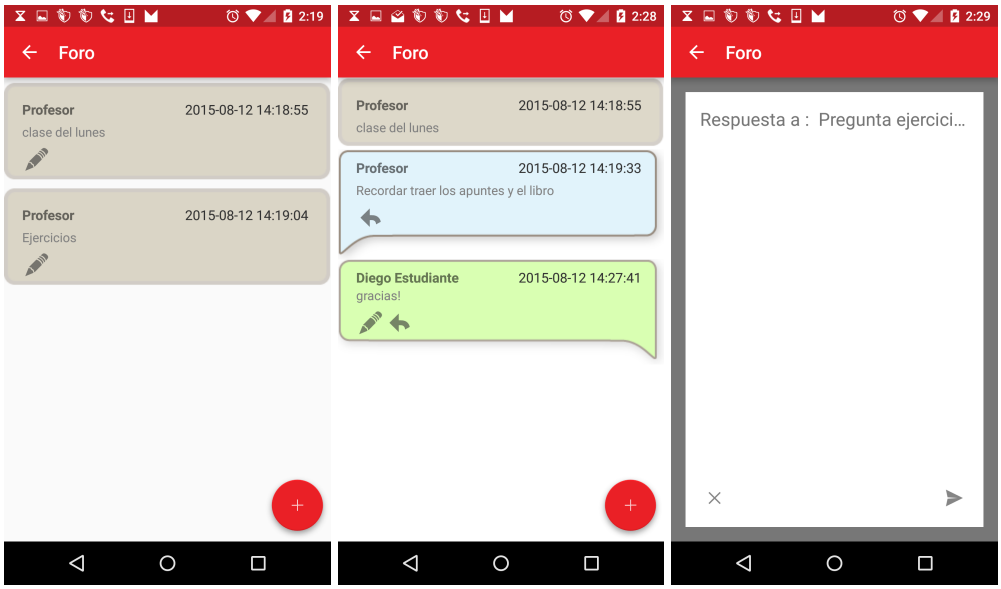


Fig. B.5 Vista foro

B.1.6 Usuario

En esta pestaña se pueden configurar los datos personales del usuario. Aquí el usuario podrá cambiar de nombre o contraseña. También en esta pestaña es posible eliminar los registros de asignaturas o modificarlas, cambiando su nombre o fotografía.

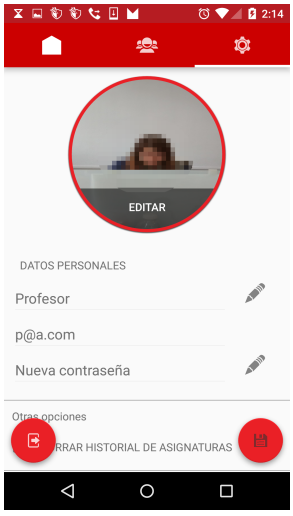


Fig. B.6 Vista usuario

B.2 Aplicación alumno

B.2.1 Vista login

La aplicación empieza con la pantalla de login. Donde el usuario podrá introducir sus credenciales y tendrá la opción de recordarlos para un auto-login.

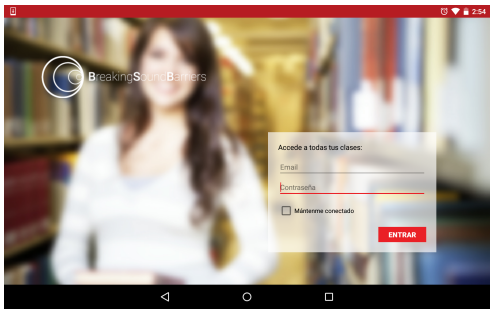


Fig. B.7 Vista inicial

B.2.2 Vista principal

En la pantalla de inicio se encuentran las últimas acciones del alumno (capturas de pantallas, notas, etc.), así como unas tarjetas que explican las diferentes funcionalidades de la aplicación.

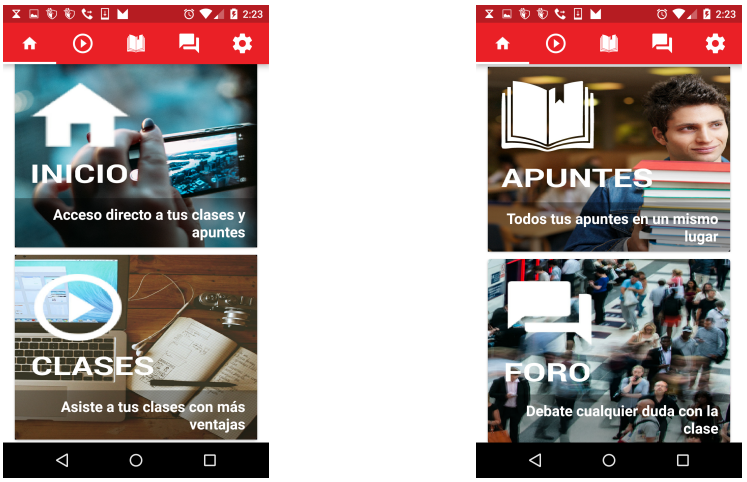


Fig. B.8 Vista principal alumno

B.2.3 Pantalla de streaming

En la siguiente pestaña se muestran las clases que actualmente se encuentran abiertas. En el caso de que la clase haya sido cerrada con contraseña, será necesario que el alumno la introduzca para poder entrar.

La pantalla streaming contiene la vista donde podremos seguir la clase del profesor. Además, existe una parte dedicado a la creación de apuntes. Para crear más de una hoja de apuntes, simplemente hay que realizar scroll hacia derecha o izquierda sobre la libreta.

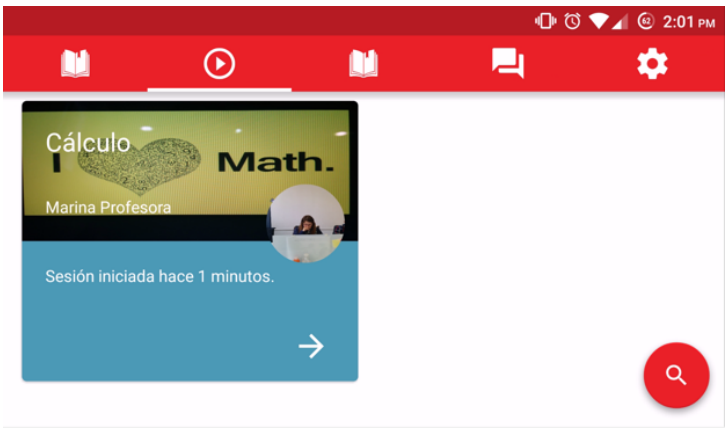


Fig. B.9 Vista streaming del alumno

Para poder realizar capturas de pantalla se debe pulsar el botón donde se encuentra un icono de cámara y automáticamente tendrá la captura guardada para visionarla posteriormente. Para realizar preguntas en tiempo real al profesor, se debe pulsar el botón pregunta.

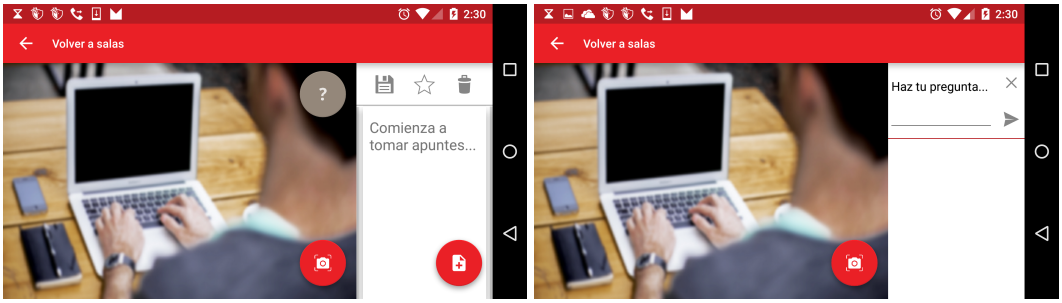


Fig. B.10 Vista streaming alumno

B.2.4 Repositorio

En esta vista se pueden encontrar todas y cada unas de las notas o capturas de pantalla que se han realizado durante una sesión de streaming. Al igual que la vista de foro, al entrar en ella se mostrarán las asignaturas a las que hayas asistido en alguna sesión de streaming. Aunque el usuario no haya generado contenido durante esa sesión, siempre aparecerá al menos la trascripción del profesor. La organización de los documentos se agrupa por fecha y sesión.

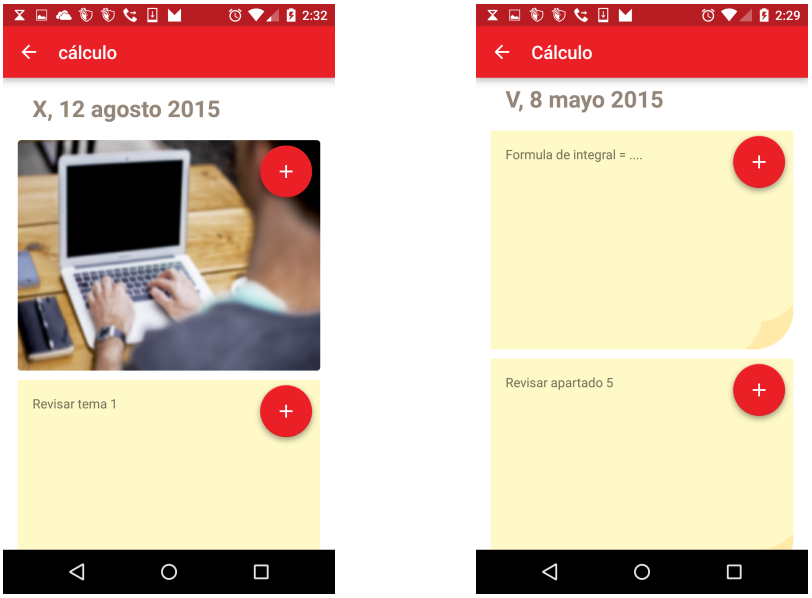


Fig. B.11 Vista repositorio

B.2.5 Foro

El foro se organiza por asignatura, que a su vez se agrupa por hilos, y dentro de estos, comentarios por hilo. Una vez un alumno ha entrado en una sala, ese alumno queda ya adscrito a la asignatura, por lo que podrá entrar en el foro de discusión para realizar o contestar preguntas. Dentro de cada asignatura, aparecerán los diferentes temas de conversación. Aquí el usuario puede crear nuevos hilos o cambiar de nombre los existentes creados por él. Y dentro de cada hilo, el usuario podrá crear nuevos mensajes, contestar a otros que ya existan o editar los existentes creados por él.

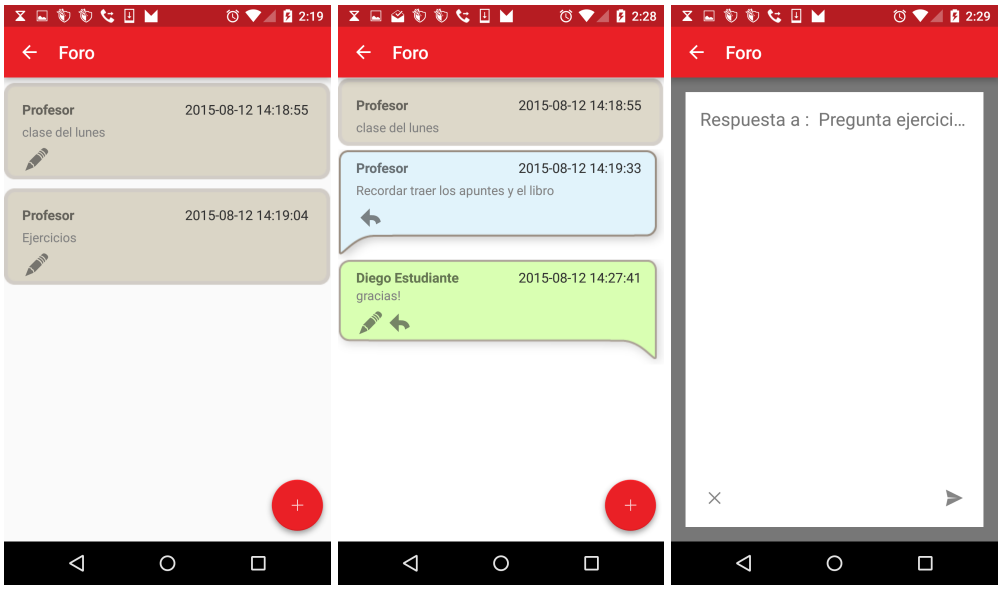


Fig. B.12 Vista foro

B.2.6 Usuario

Desde la pantalla de configuración se pueden editar los datos personales del alumno. Aquí el usuario podrá cambiar de nombre o contraseña. También tiene la posibilidad de cambiar diferentes configuración de vista.

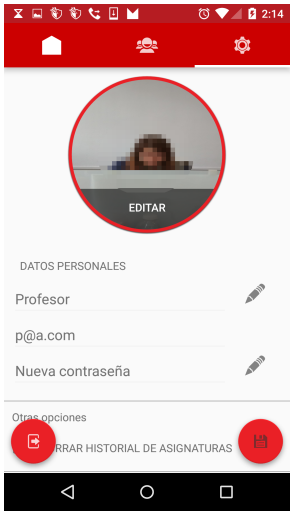


Fig. B.13 Vista usuario

Anexo C

Manual de instalación

Este anexo tiene como finalidad definir los requisitos técnicos necesarios a la hora de la implantación del sistema en una universidad, así como describir los pasos a realizar para la instalación y configuración de los componentes necesarios para el correcto funcionamiento de la aplicación.

C.1 Requisitos

Antes de comenzar la implantación, es necesario disponer de ciertos elementos esenciales en la Universidad, que, de no tenerse, habrán de adquirirse o contratarse. Estos son:

- Red Wi-Fi: mínimo a 54 Mbps.
- Dispositivos tablet: de las características acordadas en el capítulo 3.
- Un dispositivo para la aplicación del profesor.
- Ordenador: portátil o de sobremesa, indiferentemente. Como mínimo ha de reunir las características siguientes:
  - Dual Core a 1.8 GHz con 4GB RAM y 100 GB Disco duro.
  - Sistema operativo: Windows 7 en adelante o Mac OS 10.10 en adelante.

C.2 Configuración del servidor

El back-end de las aplicaciones está desarrollado sobre el framework Play, un entorno de desarrollo web basado en Java y Scala, base de datos MySQL y entorno de desarrollo IntelliJ IDEA. Para poder ejecutar todas las instrucciones que se indican en este manual se debe usar el Terminal de Mac OS X.

Para abrirlo, tenemo que ir a Finder > Aplicaciones > Utilidades > Terminal.app.

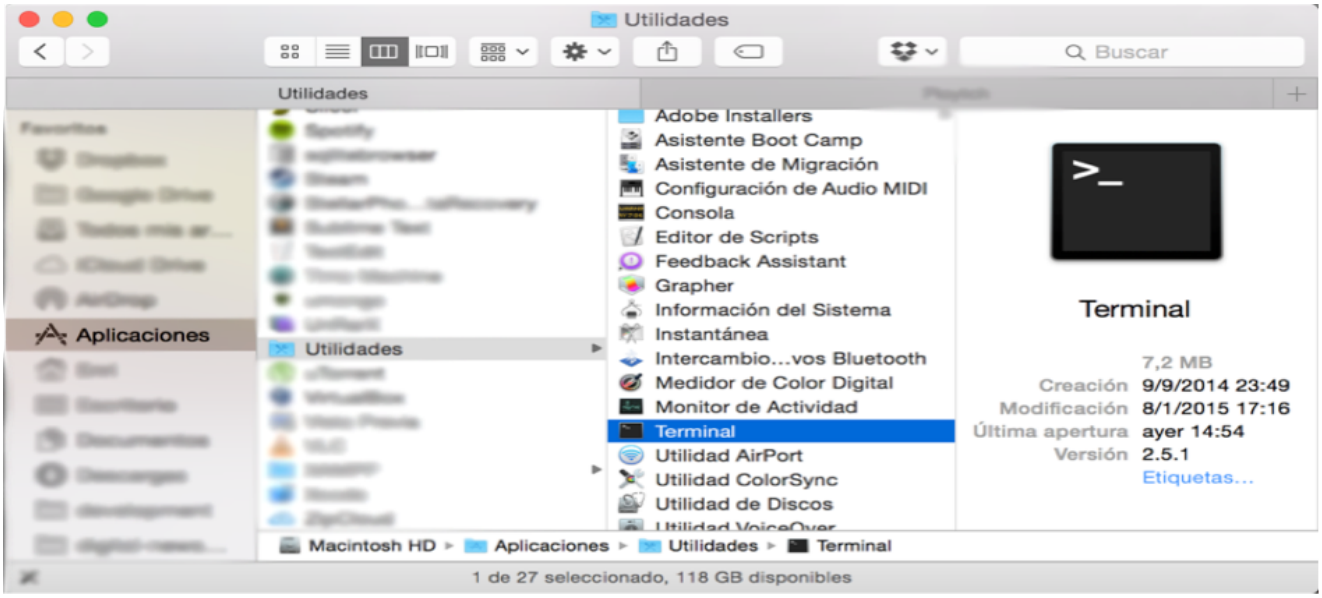


Fig. C.1 Terminal

C.2.1 Play Framework

Descarga de Play Framework versión 2.2.6 disponible en https://www.playframework.com/download.

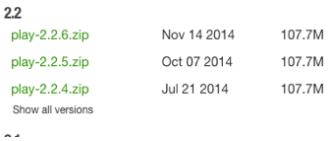


Fig. C.2 Descarga Play Framework

Una vez descargado y descomprimido sólo en necesario añadir el Path a tu entorno de desarrollo por terminal, para que esté accesible desde cualquier lugar.

```
#Setting path for play framework

export PATH=$PATH:/Applications/play-2.2.6/
```

O mediante Terminal :

```
sudo apt-get install unzip

wget http://downloads.typesafe.com/play/2.2.6/play-2.2.6.zip

unzip play-2.2.6.zip

sudo mv play-2.2.6.zip

sudo ln -s /opt/play-2.2.6 /opt/play2_2_6

sudo ln -s /opt/play2_2_6/play /usr/local/bin/play
```

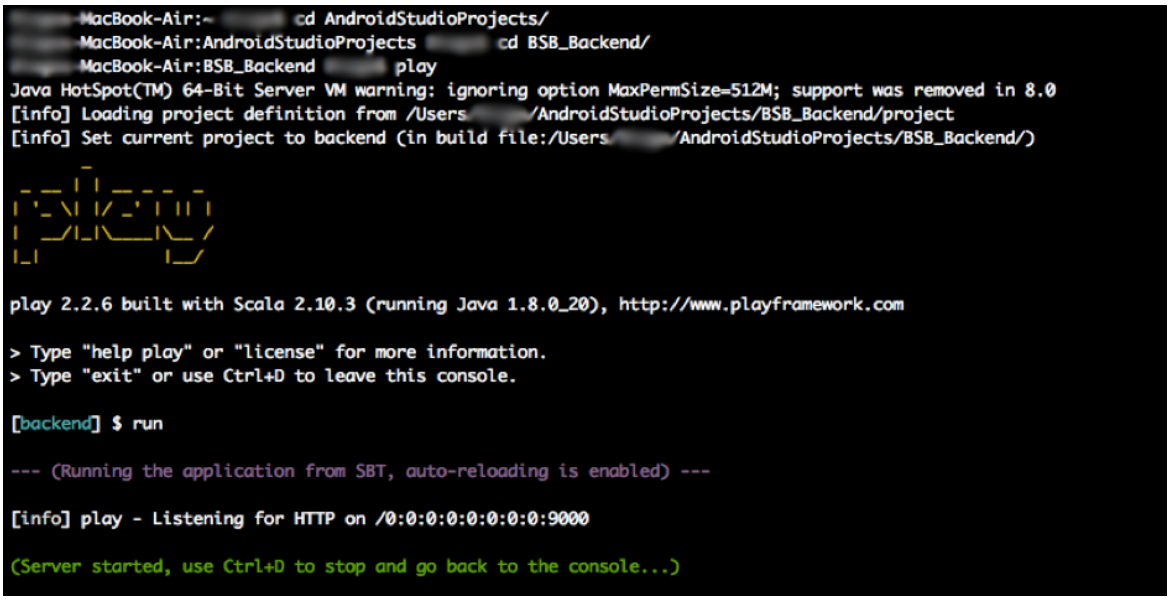


Fig. C.3 Terminal Play Framework

Ahora entrando en la carpeta donde se encuentra el proyecto con tan sólo ejecutar el comando “play”, entramos en la consola de play. Para ejecutar el proyecto ejecutar “run”. El servidor estará disponible en http://localhost:9000

C.2.2 IntelliJ IDEA

IntelliJ Idea 14 disponible en https://www.jetbrains.com/idea/download/.

Fig. C.4 Descarga IntelliJ

A la hora de instalar IntelliJ, siga los pasos de instalación y además deberá de instalar el plugin para play. Para ello diríjase a preferencias, plugins y busque por “play 2”.

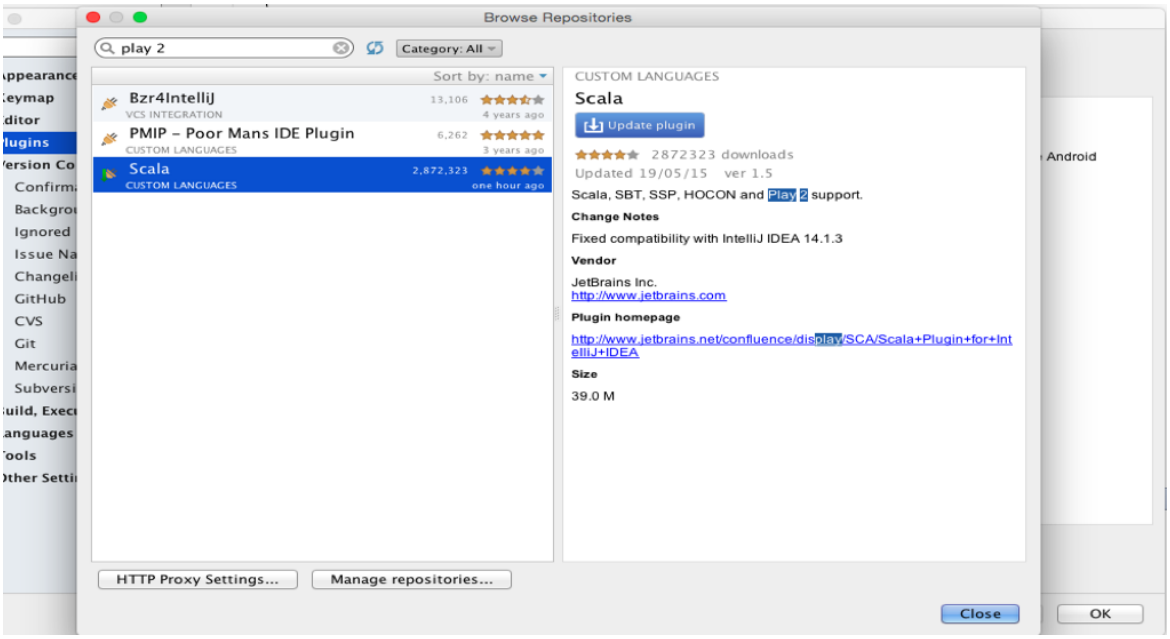


Fig. C.5 Configuración IntelliJ 1

Para importar el proyecto siga los siguientes pasos:

1. Seleccione importar proyecto con modelo SBT.

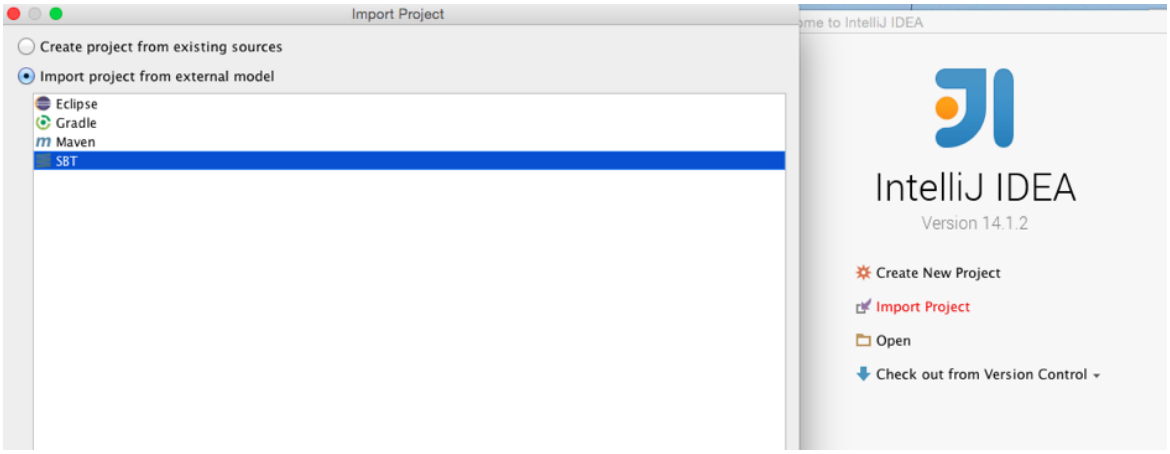


Fig. C.6 Configuración IntelliJ 2

2. Seleccione todas las opciones para su correcta configuración.

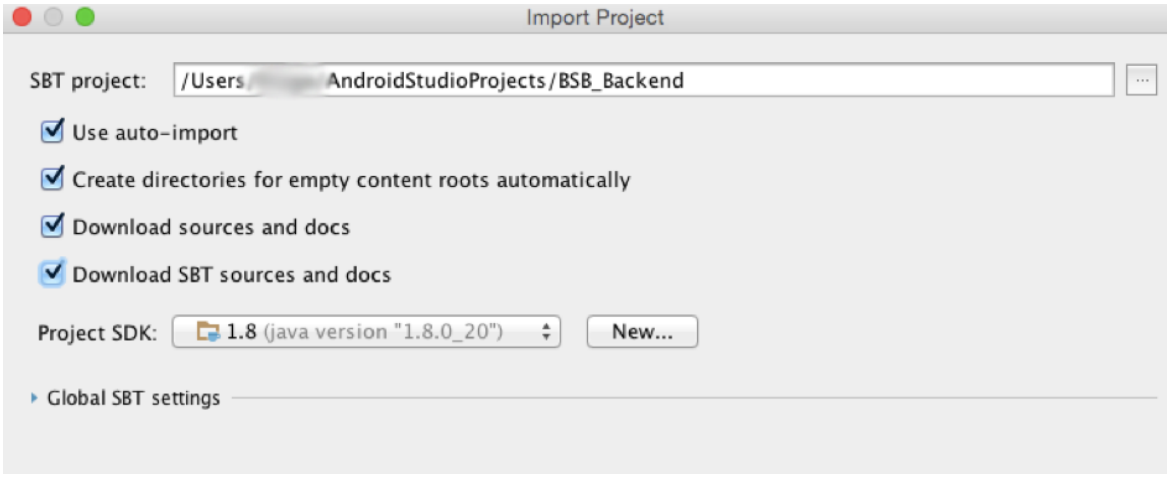


Fig. C.7 Configuración IntelliJ 3

El proceso de indexado y actualización de bibliotecas puede llevar un tiempo considerable. Para la configuración del servidor únicamente se ha de modificar el archivo application.conf ubicado en la ruta relativa a la raíz del proyecto siguiente:

`/BSB_Backend/conf/application.conf`

En ella se debe configurar el usuario y contraseña de la base de datos.

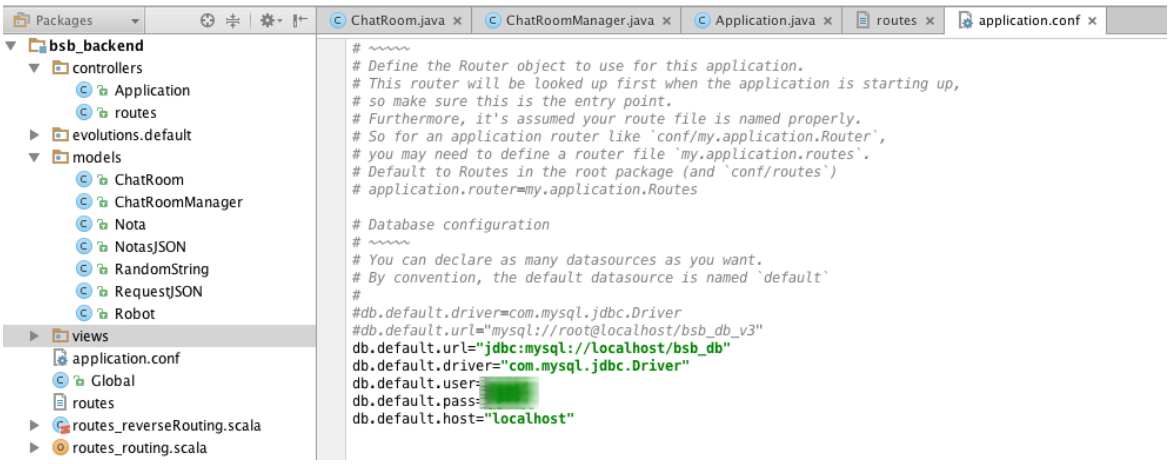


Fig. C.8 Configuración IntelliJ 4

C.2.3 MySQL

Descarga de MySQL desde https://dev.mysql.com/downloads/mysql/

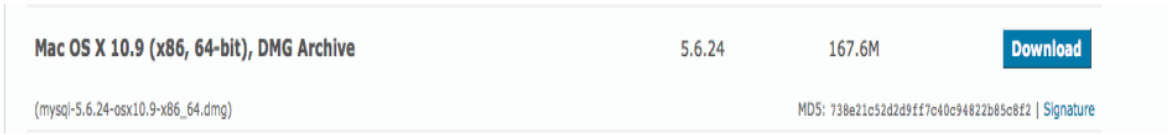


Fig. C.9 Descarga MySql

```
#Setting path for MySQL

export PATH=$PATH:/usr/local/mysql/bin
```

Entre en MySQL y cambie la contraseña o cree un nuevo usuario y añada una base de datos:

```
./mysql -u root -p

mysql> set password for 'root'@'localhost'=password('xxxx');

mysql> create user 'myuser'@'localhost' identified by 'xxxx';

mysql> create database bsb_db;
```

Inserte la base de datos `bsb_db.sql`:

```
mysql -u USUARIO -p -D bsb_db <bsb_db.sql
```

C.2.4 JDK

JDK de Java desde http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html.

O mediante Terminal:

```
sudo apt-get install openjdk-7-jdk

apt-cache search jdk

export JAVA_HOME=/usr/lib/jvm/java-7-openjdk

export PATH=$PATH:/usr/lib/jvm/java-7-openjdk/bin
```

C.2.5 Git

Por último instalar Git para descargar el repositorio:.

```
sudo apt-get update

sudo apt-get install git
```

El repositorio del proyecto se obtiene desde Git mediante.

```
git clone https://USUARIO@bitbucket.org/USUARIOCONELREPOSITORIO/REPOSITORIO.git
```

C.3 Configuración de las aplicaciones Android

Las aplicaciones Android están desarrolladas sobre la plataforma Android Studio.

C.3.1 Android Studio

Disponible su descarga en https://developer.android.com/sdk/index.html.

Dentro del SDK manager es necesario tener descargado los siguientes elementos para su correcta compilación, ya que estas son las versiones para las que se ha desarrollado la aplicación.

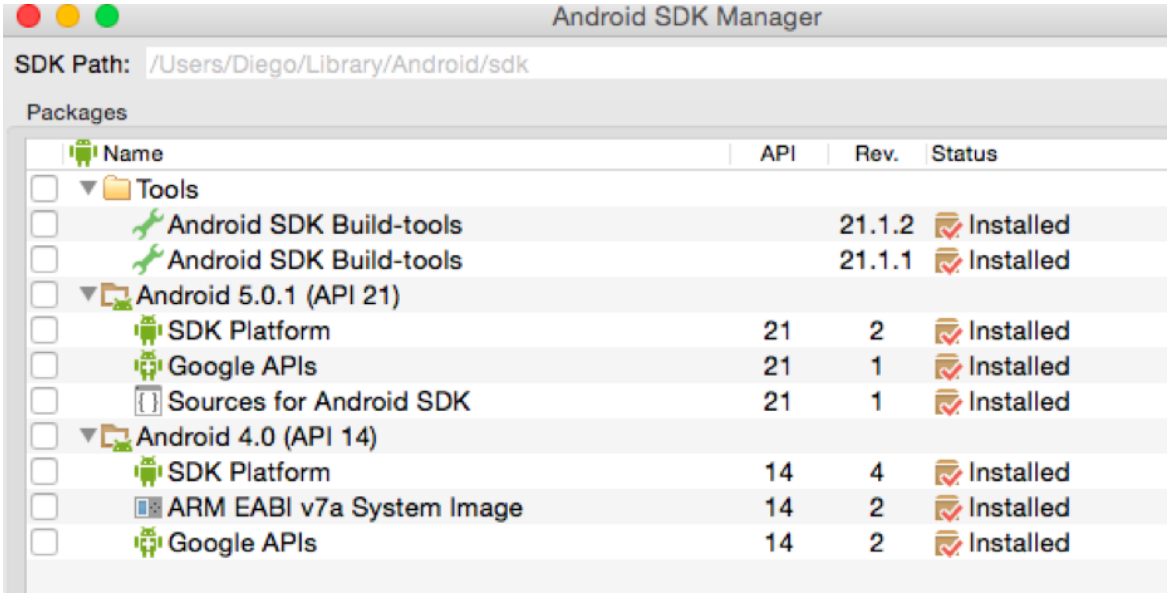


Fig. C.10 Android Studio

Para importar el proyecto seleccione abrir un proyecto existente de Android.

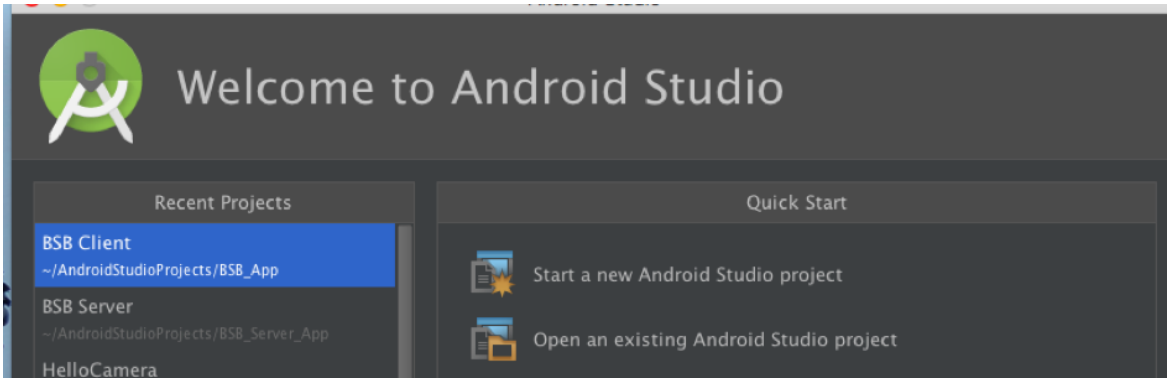


Fig. C.11 SDK Manager de Android Studio

C.3.2 Configuración

Para la configuración de la aplicación Android únicamente se ha de modificar el archivo *strings.xml* ubicado en la ruta relativa a la raíz del proyecto siguiente:

`/app/src/main/res/values/strings.xml`

En dicho archivo se ha de editar la última línea, donde se podrá encontrar una estructura similar a la siguiente.

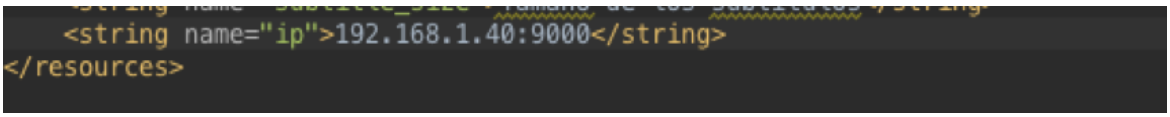


Fig. C.12 Documento XML *strings.xml*

La nueva entrada ha de seguir el patrón indicado a continuación:

`<string name=''ip''>url:port</string>`

Donde url ha de sustituirse por la dirección del servidor en internet, o su IP de acceso en su defecto, y port por el puerto de escucha configurado anteriormente en el servidor Play. Prestar especial atención al carácter dos puntos (:) que ha de insertarse entre los datos de url y port. Ponga su dispositivo Android en modo desarrollador e instale la aplicación dando al botón de play. Es posible que necesite actualizar los drivers de su ordenador para reconocer al teléfono.